Hughes El Segundo Employees Association
Atari Computer Enthusiasts
Reprints of Exchange Newsletter Articles
December 1988 Richard L. Reaser

	Title of Article	Author's Last Name	Author's First Name	Newsletter	Issue Date	Remarks
2	3.5 Tandy Disk Drive Buffer Circuit	Gratzer	Rich	PSAN	11/00/88	Schematic.
3	Box Project, The	Hoffman	George	STATUS	10/00/88	Clock, Slave, Cables, Daisy Boxes.
5	Desktop Publishing	Forbes	Donald	JACG	10/00/88	General article.
7	Eight Tips	Iken	George	HACE	10/00/88	Graphic, Dup Prog Lines, Graphic, Flash, KeyCode,Dir, Traps
10	Eight Tips	Iken	George	HACE	11/00/88	All about Scrolling.
13	Genlock, Proffessional Desktop Video	Moes	Bill	Current Notes	11/00/88	Description.
14	Guilty in Computer "Virus" Case	Deman	Les	CPAUG	10/00/88	News article on subject.
15	Parallel Disk Interface for XL/XE	Woolley	Bob	SLCC Journal	11/00/88	Schematic.
17	Personal Pascal on the ST (part 2)	Machiaverna	Paul	JACG	10/00/88	How to article.
19	Small Articles, Atari's	Russek	Joseph	Current Notes	11/00/88	Star, Roll, Rainbow, GTIA
21	SpartaDos Cartridge, The	Joins	Kieth	Mile Hi	11/00/88	Thorough review.
26	Turbo-816	Editor	Editor	MAAUG	11/00/88	New product by Data Que
27	War Inside the Computer	Small	David/Sandy	Current Notes	11/00/88	Complaints about Pascal
31	Robot & Hard Disk Info	Ed				
32	Local Surplus Ads	Ed				
33	DAT adapted for Data	Gutenl	Fred	IEEE	1/00/89	Another big storage method

STAR NX-2400

A New Multi-Font 24-Pin Printer

Reviewed by Burley Kawasaki, S*P*A*C*E

I had been looking for another printer ever since my old Gemini 10X started going flakey on me. There were basically two qualities that I was looking for in a new printer. First, I wanted it to have a good letter quality mode and print high density graphics in 300 dpi or better. Secondly, I didn't want to pay an arm and a leg. These two conditions may at first seem mutually exclusive, since most 300 dpi printers cost well over $1500. There are two exceptions: the new Hewlett-Packard DeskJet and certain lower priced 24-pin printers. However, even the DeskJet was a little more than I wanted to spend; I was hoping to find a printer for under $400.

I ended up purchasing the new Star Micronics Multi-Font NX-2400 printer. It is a 24-pin printer with a hex-density graphics resolution of 360 dpi. I love the printed output when in LQ mode! No more "dotty" letters! A sample of text quality is given following this review. In draft mode it prints 142 characters per second, 47 cps in letter quality mode. While this isn't extremely fast for a modern dot matrix printer, it's fast enough for my purposes. The NX-2400 comes with a 7K print buffer, expandable by an optional RAM card. It uses a fabric ribbon cartridge that is quick and easy to replace. The NX-2400 will emulate the printer commands and character sets of both the Epson LQ-800 and IBM Proprinter X24.

Standard features include both friction and push tractor feed mechanisms, as well as a semi-automatic sheet loader. A fully automatic sheet feeder can be purchased separately if desired. It has a paper-park feature which automatically rolls fanfolded paper out of the way; the user can then insert single sheets. When you are done printing on single sheets, the NX-2400 will roll the fanfolded paper back into place. There is no need to remove the tractor feed mechanism or fanfolded paper.

The NX-2400 has four letter quality font styles: Courier, Prestige, Orator and Script. Additional fonts can be added with font cards. These fonts can be printed in any number of pitches (10, 12, 15, 17, & 20 cpi) and sizes (triple width, quadruple height, etc.). Other print options can also be set: outline, shadow, underline, overline, italics, double-strike, emphasize, subscripts and superscripts. The Courier and Prestige LQ fonts can also be proportionally spaced. It supports standard and IBM character sets, 14 international character sets, and user downloaded characters.

All font style and pitch options can be set from the front control panel. Other features accessible from the panel include paper-parking, micro alignment, buffer clearing, and hexidecimal dump. A special "lock" mode can be engaged which prevents software from interfering with the style and pitch selected from the control panel. A quiet mode can be set, which causes the printer to make two passes per line. In reality, I didn't find this mode to be that much quieter than normal printing.

The only complaint I have is with the mail-order company that I bought it from. Not yet having read the article in the September PSAN, I decided to order through Lyco Computer. To be fair, they did admit that they were out of NX-2400's. Other places I called didn't have the printer either, because the NX-2400 was so new. However, Lyco said that they were getting a shipment in shortly (they implied within a week), and promised to ship it within 24 hours of receiving the printer. I didn't get it for another three and a half weeks! Lyco did have a good price ($309 + shipping), and I didn't really need the printer immediately. Still, the unexpected delay was pretty inconvenient. Otherwise, I have been very pleased with Star's new printer. It has done everything that I wanted it to do, and much more. It is packed with powerful features at a reasonable price. The NX-2400 is very similar to the NEC P2200, but has greater font flexibility and seems to have better construction and design.

****

11-88 ②

3.5" TANDY Disk Drive Buffer Circuit

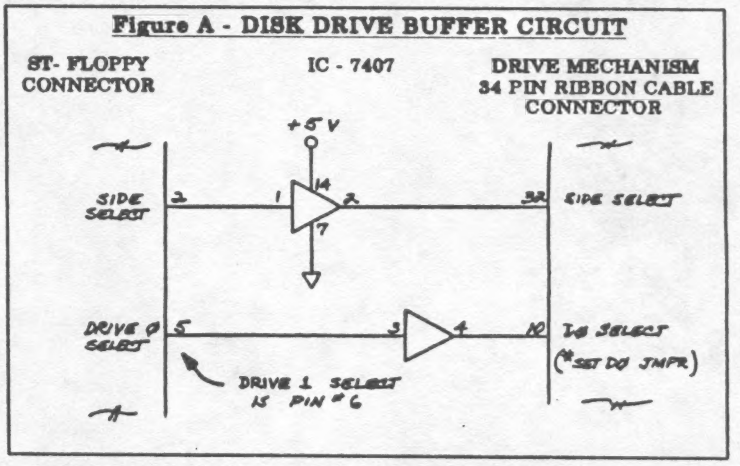
Hardware Tips by Rich Gratzer, S*P*A*C*E

Here's a quick note for those of you who are currently modifying the $99.00 TANDY Disk Drives to use on your ST or 8-Bit computers *(Ed., "Use A $99 3 1/2" 1 Meg Drive On ANY Atari", PSAN, Oct '88).*

The signal inputs on the TEAC drive mechanism have 1K ohm pull-up resistors connected to the +5 volt power. This requires the output drivers in the ST to sink about 4.9 milliamps of current to provide a logic 0 enable signal. If you add another 1.6 milliamps to allow for any type of TTL gate input, the worst case current requirement comes to 6.5 milliamps. Unfortunately, the ST's designers left out the drivers for three of the output signals: SIDE SELECT, DRIVE0, and DRIVE1. These outputs are rated at 1.6 ma. by General Instruments, and 2.3 ma. max by ATARI!? Basically, this means that the ST's Sound Chip outputs are being overloaded by approximately 300%! This causes the output logic low voltage level to rise across the output transistor (inside the Sound Chip), generating more power or heat which may eventually or immediately destroy the output stage of the IC... costing you lots of money! I've heard that the Sound Chip outputs have a tendency to be non-forgiving so be warned and be careful!

The circuit in Figure A depicts a 7407 HEX NON-INVERTING BUFFER IC. This chip has no trouble at all driving the input signals shown. I've installed one in our club's Tandy drive and it works just fine.

Figure A - DISK DRIVE BUFFER CIRCUIT



Those of you who intend to put this drive mechanism inside an older 520-STFM will need to acquire a 10K ohm resistor network to replace the existing 1K ohm network on the circuit board. This will reduce the input current requirement to an acceptable level. There are also two other alternatives you could try in a jam. The first possibility is to simply clip the pull-up resistors (of concern) between the 1K ohm resistor network and the circuit board. The ST may be able to drive the lines sufficiently high without any pull-up resistor at all, but I couldn't say for sure. The last alternative is to cut the circuit board trace and install a 1/8 watt, 10K ohm resistor in series between the signal trace and the pull-up resistor. The new resistor would have to be tack-soldered to the board across the cut trace.

In any event, none of these four modifications is a job for a novice. If necessary, have someone qualified do the job for you. Dave at BUTLER's is a fine technician and he is already aware of this problem. Give them a call!

Good Luck, Rich

P.S. When using the floppy disk connector pinout diagram in the back of the ST manual, be sure to remember that the pin numbers are reversed (swapped left to right) from what you'll see when looking at pin-end of the drive cable connector.

****

capacitor @C115. Lastly, text resolution can be improved by snipping off one end of capacitor @C60. It's enough improvement it can be noticed.

I hope this little article has helped you 1200XL owners. Look me up at the GKAUG (GREATER KALAMAZOO ATARI USERS GROUP) BBS (616)657-2665.

STATUS 10-88

THE BOX PROJECT

by George Hoffman, (The Captain)
Project by David Bunch & George Hoffman

There comes a time in every 8-BIT computer enthusiast's career when it is said, "Wouldn't it be nice 'if'....." This project hopes to supply an answer for some of these 'ifs'.

Some of these ideas were ours alone and some had their basis in Analog Magazine, under the 'Bits and Pieces Dept.', by Lee Brillant. They have been modified to make them a little more user friendly than the original project outlined in the various articles. All of these boxes plug into the joystick ports or serial port and require no modification to the computer. The software for each box will be carried on STATUS RBBS, 884-395-3985, in the 8-bit library, under applications and listed as "CLOCKn", "VOICEn", etc.

Here is what we have to offer.

THE DAISY BOX

A box with four 13 pin serial sockets in it, with or without a 13 pin serial cable attached. When a device is disconnected from the system, you no longer have to worry whether the daisy chain is broken. The DAISY BOX requires no power.

THE CLOCK BOX

A box that plugs into joystick ports 1&2 or 3&4. Once it is set with the correct data, it keeps the time, date and day of the week, whether it is plugged into the computer or not. When it is plugged in and the computer is on, it draws its power from the machine, otherwise it uses an internal source for power. We estimate the internal source will drive the clock for a week without a recharge from the computer, which only takes seconds to accomplish. In addtion, the box has a 1/8" (3.5mm) jack in the side for a 3 volt DC supply, (2 AA batteries) which will allow the clock to run as long as the supply lasts (approximately 1 yr). Both the internal backup and the 3 volt DC supplies are isolated from the computer and are not used if plugged in and the computer is on. The software that reads the data is in basic (just a few lines) to be included in your basic programs and in ML if you wish it to remain resident from bootup. Future plans call for a patch, to various DOS's, to be supplied that will time/date stamp files as they are saved.

THE VOICE BOX

A box that plugs into joystick ports 1&2 or 3&4 and allows the computer to speak words that you program it to say, using a text to voice editor. The voice can be output to an external preamp and speaker or to the TV/Monitior speaker. This box could be used in conjunction with the CLOCK BOX to vocalize the time when requested. Another use could be to vocalize the text on a screen for childrens educational programs. An example would be a flash card program. If there is sufficient interest, a vocal dictionary will be maintained and listed on the STATUS RBBS. The VOICE BOX is powered by the computer.

THE SLAVE BOX

This box connects two computers via the 13 pin serial ports and allows you to use your extra 400/800/XL/XE as a programmable slave computer. This slave machine can in turn be connected to the printer via the joystick ports using 8 bit parallel transfer. This would make the slave computer a 47k printer buffer and it would store about 20 pages of text, freeing up your main machine for other purposes while the slave prints out the buffer. In theory it is possible to use the same box in any number of configurations. Some examples would be a modem buffer or an extra computer running a security program and reporting back to the master machine. These are all software programmable options and are presently in the works. The SLAVE BOX is powered by the computer.

CABLES

Finally, we have 13 pin serial cables available in lengths of 1, 3 and 6 feet. Six and nine wire joystick cables are available, on request.

All of these BOXes work on the 400/800/XL and XE machines, as is. If an XL or XE is used as the slave, the SLAVE BOX requries some additional software and cable changes to work properly, in addtion the extra ram is available for use. This is presently in development and may be available by the time of this printing.

Prices:
13 pin cables:
1 foot $3.95
3 foot $5.95
6 foot $7.95 With audio input, add. 1.25
Daisy Box:
w/1 foot cable $18.95
w/3 foot cable $21.95
w/6 foot cable $24.95
Clock Box: $29.95
Ext. power supply $2.15
Voice Box: $39.95
Ext. output Pre-amp/speaker $12.50
6 foot audio cable $2.40
TV output- see Cable list
Slave Box:
with 3' cables $20.95
with 6' cables $26.95
slave/printer 8 bit interface cable $14.95
48k 400's- $59.95
48k 800's $79.95

Please direct all inquiries to THE BOX

PROJECT, P.O. Box 331, Portsmouth, Va., 23705, Phone 804-483-1378. STATUS RBBS Phone# 804-495-3985

----*----

WANDERER

3-D ACTION GAME
PYRAMIDE (French)

Computers: 520, 1040ST Color monitor. Joystick required.

Reviewed by Bruce Finkelstein

While wandering (no pun intended) through the local software/electronics boutique, some marked down ST software caught my eye. I had not purchased entertainment software (read that GAMES) in quite a while and was intrigued by the colorful plastic box with a space battle scene on it. The blurb on the back touted it as "a game in space and at least three dimensions". The price was right and I was curious about 3-D programs.

When I opened the package I found the program disk, a 4x6 instruction pamphlet, and two pairs of cardboard red and blue glasses.

The 3-D effect is not the high tech supplied by "Stereotek" glasses but the same two colored glasses used in movie theaters in the fifties. The graphics are drawn in two colors, offset in order to produce the 3 dimensional effect through the bicolor glasses. There is a good explanation of the 3-D, and some suggestions for improved viewing in the instruction book.

I booted the program, and after getting used to the 3-D effect, played for a while. After not getting too far in the game I decided to read the instructions.

The premise of the game is familiar, collect points (in this case, cats), advance in rank, and defeat the final enemy. The venue is very familiar to those of us who started computing on Atari 800's. The game is remarkably like the classic, Star Raiders. It is played going from sector to sector, destroying enemy ships along the way. During the combat phase you are presented with a "view screen" and two radar displays showing relative bearing from above and behind positions. You start with a number of shields, energy, and two poker cards. The interesting departure from the usual "zap the aliens" scenario is the game within the game. The planets hold poker hands, and you act as a courier, trading cards for cats. The amount received for the cards you trade is dependent on how much the hand held is improved. This introduces an element of strategy, as it is important to plan your travels to maximize profit, and minimize energy expenditure.

There are other means of advancing to the final confrontation including collecting jokers from the black holes and assembling 5 aces. You can save and resume up to 10 games should you wish to complete a game at a later time.

This game is by no means easy, and is more of a novelty than an addiction. If you are curious about 3-D and would like to add one to your game collection, this program might be what you're looking for. If you are looking for an addictive arcade style game that you can play for days, you might have to look elsewhere.

----*----

TELE-ETIQUETTE

A Personal Primer for Milder Modem Manners
By Bob Womack for STATUS

OK, OK. I know some of you have been telecommunicating for years. You've called BBS's all over these United States. You may have every possible kind of log-on sequence memorized so you can interpret within milliseconds the initial transmission from any computer (which usually looks like random ATASCII hash to the rest of us). You've even been called BACK and madly bleeped at by a No-Change (IBM) BBS. So you've really got the technical stuff DOWN. Just about any personnel manager would say, "Great! You've got half the job worked out. How about the other half: the interpersonal one?".

Good question. Why? Because telecommunication doesn't consist of computers talking to each other, it's people communicating to each other using the computer environment as a MEDIUM. And communication between humans can be a pretty complex process. Think about it. In person, you can hear the words a person speaks, see his body language, and read the feelings in his face. Even on voice telephone, you not only hear his words, but HOW he says them. When you are communicating by computer and modem, all you have to carry your meaning are the characters you type and the manner in which you type them.

You have two "audiences" to speak to on a BBS: your fellow BBS users and the equipment operators (usually System operators or "SysOps"). Let's look at how you can best communicate to each of these.

First, the other users: The most direct impact you have on them is through the messages you leave on message bases. When you leave a message, your choice of words, the form of your messages, and the manner of your replies to other's messages, all communicate something about yourself to those watching. At the most shallow end of the issue, taking the time to organize your letter will communicate something of your personal organization. For example, when you post a message on a new subject, come up with an interesting and informative title for your message. I've been on boards where the original message title from last year ("Yea, a new message base!") still adorned every message because no-one would take the time to POST a message instead of REPLY to an existing one. The "Title Scan" function was worthless on this board for the obvious reason that every title was the same.

DESKTOP PUBLISHING

10-88 by Donald Forbes - JACG

Desktop publishing has now become a reality for many business word processing users since Version 5 of WordPerfect is now available for the business PC. They will be in a position to publish high-grade newsletters using their laser printers and current resources. They will also be able to generate professional looking documentation for all their software with a minimum amount of effort.

WordPerfect addressed our user group at the September meeting. Bill Hawes, sales representative for New York and New Jersey, gave us a slide presentation and donated a copy of the ST version as a door prize.

WordPerfect claims that they are outselling the competition two to one. They have brought out WordPerfect for the Atari ST computer based on Version 4.1 for the MS/DOS machines. I expect that once they establish a market for the Atari machines, they will come out with a desktop publishing version that will take advantage of the superb graphics of the Atari ST.

The ST is a big seller in Europe, both to the home and business markets, where buyers are not beholden to the Big Blue image of IBM. Atari has also repackaged their old 800 eight-bit machine, not as a computer, but as a GAME machine. It lists for about $200 and sells for about $150, with a detachable keyboard, and the option to attach a disk drive (and thereby turn it into a computer!). The dealers refused to handle the Atari 800 as a computer, but they are more than happy to market it as a game machine, complete with joystick and light gun. The product is a success (it sells itself with little or no advertising).

If I am not mistaken, someone at the last user group meeting said that Atari is now among the Fortune 400 corporations in terms of profitability. The European market is where the money is.

Desktop publishing traces its history back to 1455, the date of the printing of the earliest book printed from movable type, the Gutenberg Bible (some 40 copies exist today, including copies at the Huntington, Morgan, New York Public, Harvard and Yale university libraries).

Johannes Gensfleisch zur Laden zum Gutenberg was born on an unknown day between 1390 and 1400. He invented a method of printing from movable type that was used without important change until displaced by the computer. The unique elements of his invention consisted of (1) a mold with punch-stamped matrices (metal prisms used to mold the face of the type) with which type could be cast precisely and in large quantities, (2) a type-metal alloy, (3) a new press derived from those used in wine making None of these features existed in Chinese or Korean printing, or in the existing European technique of woodblock printing.

Gutenberg was a inventor who placed perfection above promptness, while his business partners wanted a safe and quick return on their investment. Gutenberg's dream was to create a way of mechanically reproducing medieval liturgical manuscripts without losing any of their color or beauty of design. They sued, and Gutenberg lost: he was ruined financially on November 6, 1455.

He died in poverty and almost blind in 1468. Were he alive today, he would be a millionaire many times over.

We have come a long way since Gutenberg. His Bible had no title page, no page numbers, and no innovations to distinguish it from the work of a manuscript copyist. But he brought the written word to the public, and thus is responsible for publishing as it has been known for five hundred years.

Until now, publishing has belonged to the few: those with the knowledge, the money, and the time to engage in a complex process involving many players. The process begins with an idea. It continues with a writer, who originates the material to be published. An editor modifies it for accuracy and editorial slant. The designer gives the publication its shape. A typesetter sets the words. An illustrator enhances the ideas with pictures. A paste-up artist assembles the publication. A commercial printer completes the process. Each one of these skilled steps takes days or weeks, and must be precisely coordinated. Just managing the process can be a full-time job.

Now we have come full cycle since Gutenberg. With desktop publishing using WordPerfect or any other package, you can now perform all those complex functions and produce quality printed material in much less time and at much less expense.

Desktop publishing today is the use of personal computers to create newsletters, flyers, brochures, advertisements, catalogs and manuals, and combines layout, design and integration of graphics to create what looks like a professionally published document. Ventura Publisher and PageMaker are the best known software packages. When you use them, however, you must use two separate programs: a word processing program to create the text, and then retrieve it into a desktop publishing program, both of them powerful but complex and difficult to master.

WordPerfect needs only one program. You can import drawings and illustrations from other graphics programs, and squeeze, expand, scale, rotate, move or invert them, and embed them in boxes, horizontal and vertical rules, and tables. You can add borders, captions, and gray shading. Leading, kerning (changing the space between wide and

skinny letters), and word and letter spacing are available. All this, without having to master a complex program. It is not, however, a WYSIWYG (what you see is what you get) program. You cannot just manipulate the final output on the screen, and therefore you have to plan everything carefully in advance.

Desktop publishing can be a successful hobby as well as a business. Bob Denton, one of our ST-owning members, publishes a newsletter on salt-water aquarium fish (an esoteric field about which most fresh water fish collectors know nothing).

The door prize, a copy of WordPerfect for the ST, was won by our longtime member Stefan Andrus Burr, once of Lockheed and AT&T who is now a professor of computer science at City College in New York City. Some years ago he organized a conference on the Mathematics of Networks for the American Mathematical Society. Last summer he organized a similar short course on chaos and fractals.

The centennial celebration of the American Mathematical Society in Providence RI in early August attracted 1700 people. The weekend before the meeting, the AMS sponsored an enormously successful short course on chaos and fractals. The course attracted a record crowd of about 500. The diverse crowd included many graduate students, participants from industry and laboratories, and even some high school teachers.

The audience heard presentations on such topics as the horseshoe map, chaotic attractors, Julia sets, and iterated function systems. In addition, they saw computer generated illustrations and films representing the mathematical objects explored in the course.

Now that Stefan has a copy of WordPerfect, perhaps he will give us a write-up for the newsletter of the highlights of his short course.

MESSAGE from the SYSOP

Gary Gorski - JACG

In order to stay current on this BBS, you must call at least once every 60 days. If you do not call, the sytem will automatically delete you. You then will have to **REREGISTER** to be a user on the JACG BBS... If you would like to check your EXPIRATION date, [Q]uit to main menu, then [V]iew options.

AN 8-BIT FLIGHT SIMULATOR II TOUR

(6)

WITH
SUBLOGIC SCENERY DISK #11

Dave Arlington - JACG

This article is a step by step tour of a flight from Buffalo, New York to Toronto, Ontario using an 8-bit Atari computer, Flight Simulator II, and the new SubLogic Scenery Disk #11 which covers Western New York and Southern Ontario as well as western Pennsylvania, northern Ohio, and most of the state of Michigan. This article is written especially for the Western New York Atari Users Group and the Toronto Atari Federation and has been submitted jointly to both newsletters.

If you have FSII and rarely use it due to your lack of flying ability, you will like this article since it gives step by step instructions and is Atari specific. You will be able to take off, fly and land by reading along. If you're fairly adept as FSII, you can discover some of the features and quirks of the SubLogic Scenery disks as well as seeing how they treat our area. And of course, I hope this article inspires everyone to rush out and buy Scenery Disk #11 so SubLogic will continue to support the 8-bit Ataris as they have done in the past.

If you already have Scenery Disk #11, you can follow along with this flight step by step. If not, read along anyway. When and if you acquire the disk, you can come back and fly along with the rest of us.

Load up Flight Simulator using either the disk version or the XE Game System cartridge. (Later on, I'll compare the two versions. I think you'll be suprised at which I prefer.) Insert the Scenery Disk #11B and press Control-E to log the disk in. Make sure you use disk 11B since that is the one with the Buffalo-Toronto area. , Press any key and then hit ESCape to enter the editor. The mode number should be zero. Now enter 110 for the mode number and you see the mode number changes to 10, the first of the user-defined modes.

Now, leaving everything unchanged EXCEPT the parameters I give you, type these parameters in:

NORTH: 17844
EAST: 19453
ALTITUDE: 725
HEADING: 215

When you are done, press Control-S to save this mode in memory. That way, if you happen to crash, you will be right back where you started. Hit ESCape to exit the Editor. The disk loads some information and there you are... Check your heading to make sure your compass (on the instrument panel, not the one up on the windshield) reads a heading of about 215-216. If so, fine. If not, hit Control-= to get things straight.

HACE 10-88

EIGHT TIPS

by George Dhen
Houston Atari Computer Enthusiasts

Some of you may have noticed that the mid-October file in Database 10 of the HACE BBS (713-458-9923) had a text file of 8-bit programming tips from the "01 HACKERS Atari User's Group". These included a beginning BASIC tutorial, an advanced BASIC tutorial, and a SPARTADOS tip. We hope to put up more items of that nature in Database 10, so keep an eye out for them when you are logged onto the HACE BBS.

With that message out of the way, on with the column. The following 8 bit tips are from a variety of sources. Hopefully they can be of use in your BASIC programming. The following BASIC examples are used to demonstrate some of these tips. The BASIC programs work with just about any DOS (Atari DOS, SPARTADOS, and SMARTDOS have been tested), and are written in (and have only been tested as) Atari BASIC:

GRAPHIC DEMO:

```
10 POKE 764,255:REM 255 means no key has been pressed on the keyboard

20 PRINT CHR$(125):GRAPHICS 17:POSITION 3,1

30 PRINT #6;"HOUSTON ATARI":PRINT #6;"COMPUTER ENTHUSIASTS"

40 PRINT #6:PRINT #6;"ARE YOU A MEMBER?":PRINT #6;"y OR n":

50 IF PEEK(764)=43 THEN 100:REM goto line 100 if a "Y" or a "y" is pressed

60 IF PEEK(764)=35 THEN 200:REM goto line 200 if a "N" or a "n" is pressed

70 GOTO 50:REM a Y or N has not been pressed.

100 PRINT CHR$(125):GRAPHICS 1:POSITION 3,1

110 PRINT #6;"WRITE A HACE":PRINT #6;"NEWSLETTER ARTICLE"

120 GOTO 220:REM say thanks and end program

200 PRINT CHR$(125):GRAPHICS 1:POSITION 3,1

210 PRINT #6;"WE WOULD LIKE":PRINT #6;" TO HAVE YOU JOIN"

220 PRINT #6:PRINT #6;" THANKS!"

230 PRINT #6:PRINT #6;" h.a.c.e."

240 POKE 764,255

250 PRINT "PRESS ANY KEY TO STOP":IF PEEK(764)<>255 THEN 270

260 GOTO 250

270 GRAPHICS 0:REM end program and clear character graphics from screen
```

1. DUPLICATING PROGRAM LINES: Atari BASIC uses a "full screen" editor. Anything that is on your monitor screen is "live" and can be edited by positioning the cursor at that point and pressing another key. In "programmed mode" (ie where each line is started with a line number) you must press the RETURN key while the cursor is still on that line to make the change permanent. This means you can easily duplicate a line, simply by moving the cursor up to a line on the screen, and typing a new line number over the old line number. Press the RETURN key to save that change, and you now have two duplicate lines in memory (one with the old line number and one with the new line number). Examples of lines duplicated and then edited in this manner are found in the above BASIC program with lines 20, 100, and 200 and also with lines 50 and 60 and with lines 110 and 210.

NOTE: full screen editing is just one of many similarities between your Atari and a MS-DOS (ie IBM) computer. Having used an Atari before I got an IBM at work, it was an easy conversion to the similar IBM DOS and editor.

2. DUPLICATING AN IMMEDIATE MODE COMMAND: Immediate mode is when a BASIC statement is written without a line number. The computer tries to execute that command as soon as you press the RETURN key. You can use the Atari full screen editor to repeat an immediate mode command by positioning the cursor on any line on the screen that you want to reexecute. This can come in handy if you back up your programs as you write them (this is a VERY GOOD thing to do, by the way). For example, you ordinarily save programs in immediate mode as follows:

```
SAVE "D:FILENAME.BAS"
or LIST "D:FILENAME.LST"
```

Then you can repeat the command for a second copy as follows: After you get the READY prompt again, put in a different disk, move the cursor back up to the SAVE or LIST command, and press RETURN again. This repeats the command, and you don't have to worry about misspelling the filename.

3. GRAPHICS MODES: Graphics 0 (GR.0) is the "text" mode you use when you are typing in a program. The GRAPHICS DEMO program uses Graphics mode 1 (GR.1) to provide a different look on screen. Lines 110 and 210 illustrate the short print statements that are needed in GR.1 because this mode displays only 20 columns and 20 rows in a graphics screen plus a 4 line text window (in GR.0) on the bottom. Adding 16 to the Graphics mode statement, as done in line 20 (GR.17 is really GR.1 + 16) removes the text window, so the monitor screen is totally graphics (24 rows). Printing to the graphics screen is done by a PRINT #6 statement. A print statement without reference to device #6 results in display to the text window instead of the graphics screen. The GR.1 text window can be handy, as shown in the print statement in

line 250 where a prompt to the keyboard operator is printed in the text window.

4. FLASH ON, THEN THE GRAPHIC'S GONE: When this happens, poetry is not the first thing on my mind. If you are in GR.17 mode (no text window), a print statement without reference to device #6, immediately drops you back to GR.0 mode. This also happens if you run to the end of a program, as the READY prompt is automatically put out by the computer in GR.0 mode. What you see is a flash of the GR.17 screen, and then the familiar GR.0 screen with the ready prompt shows up. This can be avoided by sticking to GR.1 (with a text window) instead of GR.17, or by putting the program in a loop, so it does not end. Lines 250 and 260 in the above program would work with GR.17 if you removed the PRINT statement from line 250. I believe this problem (and the loop solution) exists with most (all?) Graphics modes that do not have a text window.

5. MORE ON KEY CODES PEEK(764): Last month's column had a tip on use of memory location 764 which stroes a value representing the last keypress. The Graphics Demo checks for the values 43 (Y) and 35 (N) in lines 50 and 60. A table of keypress values can be found on page 131 of "ATARI BASIC" by Richard Haskell. The values are internal codes, not related to ASCII or ATASCII, and are in no particularly recognizable order. The PEEK(764) values from "ATARI BASIC" are as shown:

A=63, B=21, C=18, D=58, E=42, F=56, G=61, H=57, I=13, J=1, K=5, L=0, M=37, N=35, O=8, P=10, Q=47, R=40, S=62, T=45, U=11, V=16, W=46, X=22, Y=43, Z=23. 0=50, 1=31, 2=30, 3=26, 4=24, 5=29, 6=27, 7=51, 8=53, 9=48, ,=32, .=34, /=38, ;=2, +=6, *=7, ==15, -=14, <=54, >=55, SPACE=33, RETURN=12, ESC=28, TAB=44, BACK SP=52, INVERSE=39, LOWR=60, CAPS=124

A base keycode value is available for 64 keys (values 0-63) as shown above. Pressing the shift key and another key, adds 64 to the base value (see LOWR and CAPS). Pressing CTRL and another key adds 128 to the base value. As stated last month, the keypress value of 764 is retained until another key is pressed...OR UNTIL...an OPEN statement or a READ statement has been executed.

6. DIRECTORY LIST FROM BASIC: The next tip is the demo program itself. The program lists a disk directory to the screen FROM BASIC. No need to go to DOS. Universal application depends on the DOS that was booted up, in that SPARTADOS can read SPARTADOS directories as well as Atari DOS and SMARTDOS directories in any density (if the disk drive can read that density). If you booted up with SMARTDOS you can read SMARTDOS directories and Atari DOS directories (in any density), but not SPARTADOS directories (which are at different sectors than SMARTDOS or Atari DOS directories). Booting up in unmodified Atari DOS can read Atari DOS or SMARTDOS directories but only in single (DOS 2.0) or single and enhanced density (DOS 2.5 or (giggle) DOS 3.0). I use this code as a subroutine in many programs. The OPEN statement in line 32004 is what does the trick. The second number (6 in this case) when OPENing a DISK DRIVE tells the computer what kind of activity will be going on. The numbers can be: 4 (read only), 6 (read disk directory), 8 (write new file), 9 (write onto the end of an existing file), 12 (read or write).

I found the original code in the book "Advanced ATARI BASIC Tutorial" page 49, by Robert A. Peck, but the same code with different line numbers is also in the earlier "Your ATARI Computer", page 263, by Ion Poole, Martin McNiff, and Steven Cook. The code has been modified for publication here to list a directory from any specified drive (0 through 9, including ram disks):

```
DIRECTORY DEMO

1 REM lines 32000-009 DISK DIRECTORY PROGRAM by G N Iken, for HOUSTON ATARI COMPUTER ENTHUSIASTS, 10/88

32000 DIM A$(20),DRIVE$(6):REM A$ is a single directory listing, DRIVE$ is a drive # followed by ":*.*"

32001 TRAP 40000:TRAP 32002:DRIVE$="D1:*.*":CLOSE #1:REM trap for illegal drive #

32002 ? CHR$(125):TRAP 40000:TRAP 32001:? "Which DRIVE for disk directory":? "Number only ";:INPUT DRNUM

32003 DRIVE$(2,2)=CHR$(DRNUM+48):REM sets drive # to a string value equal to DRNUM

32004 OPEN #1,6,0,DRIVE$

32005 TRAP 32009:REM goto 32009 instead of giving an error message when last directory entry is read

32006 INPUT #1;A$:REM read an entry from the directory

32007 PRINT A$:REM print the directory to the screen..you could open a printer or filename and print to that also

32008 GOTO 32006:REM go get another directory listing

32009 CLOSE #1:TRAP 40000:PRINT "Directory listing for ";DRIVE$:REM end program, close device, clear trap flag
```

7. MORE ON TRAPS: Last month one of the tips was on error TRAPping in BASIC programs. The DIRECTORY DEMO uses TRAPs to prevent error messages when a letter key is pressed (line 32002, DRNUM is a numeric variable), when a drive number doesn't exist (needed double TRAPping on line 32002 and line 32001), and when the last entry is reached in the directory (line 32005). Once a TRAP has been "sprung" the TRAP flag is cleared, and if another error can occur, a new TRAP statement must be made BEFORE the error occurs. But before a new TRAP statement can be set, any previous TRAPs must be cleared. TRAP clearing is done by the TRAP 40000 (TRAPs are cleared by any line number over 32768 and less than 65535), and just in case a previous

RAP has not been sprung, TRAP clearing is executed just in front of each "real" TRAP statement.

8. FORMAT FROM BASIC: Atari BASIC can perform some general input/output functions with an XIO command. The FORMAT DEMO program below uses an XIO command (254 is used for the format task). The program allows repetitive formatting (prompted all the way) in SINGLE DENSITY. Drive 1 is the only drive allowed, as written, but changing D1 in lines 32014 and 32017 to some other drive number is a simple step.

Note, documentation on XIO says that a channel must be opened to the disk drive before executing the XIO command. This is done in line 32014 of the FORMAT DEMO, and a TRAP is required, because the OPEN statement wants a filename with Atari DOS (D1: is not a legal Atari DOS filename), you TRAP at the OPEN statement and the disk drive never whirls into action. In SPARTADOS and SMARTDOS, D1: apparently is a legal filename and the disk drive does whirl up (but it wants to find filename D1: which is kind of hard to find on a blank, unformatted disk). You can press the BREAK key or open the drive door to "nudge" the TRAP, but the generic solution is to remove the OPEN statement from line 32014 (this works on Atari DOS too, despite the documentation saying it is needed). I left the troublesome OPEN statement in, so you could see what it supposedly HAD to look like.

FORMAT DEMO

```
1 REM lines 32010-019 FORMAT DRIVE 1 PROGRAM
by G N Dken, for HOUSTON ATARI COMPUTER
ENTHUSIASTS, 10/88

32010 POKE 764,255:? CHR$(125):POSITION
1,10:? "WARNING ready to format DRIVE 1":? :?
"Y continues, N stops"

32011 FOR WAIT=1 TO 400:NEXT WAIT:IF
PEEK(764)=255 THEN 32010:REM waiting for ANY
keypress

32012 IF PEEK(764)<>43 THEN ? :? CHR$(125):?
"DISK FORMAT ABORTED":? "PROGRAM IS STILL IN
MEMORY, BE CAREFUL":END

32013 ? CHR$(125):? "YOU ARE CONTINUING THE
FORMAT":? "LETS DBL CHECK, DON'T PANIC"

32014 TRAP 40000:TRAP 32015:CLOSE #1:OPEN
#1,8,0,"D1:":REM end this line at CLOSE #1
for other DOS's

32015 POKE 764,255:? "INSERT DISK, PRESS Y TO
FORMAT D:1":FOR WAIT=1 TO 500:NEXT WAIT:IF
PEEK(764)=255 THEN 32015

32016 IF PEEK(764)<>43 THEN 32019:REM abort
if keypress is not a Y

32017 POKE 764,255:? CHR$(125):?
"FORMATTING":XIO 254,#1,0,0,"D1":? "FORMAT
ANOTHER Y or N"

32018 FOR WAIT=1 TO 200:NEXT WAIT:IF
PEEK(764)=255 THEN 32018

32019 ? CHR$(125):? "PROGRAM IS STILL IN
MEMORY, BE CAREFUL":IF PEEK(764)=43 THEN
MEMORY, BE CAREFUL":IF PEEK(764)=43 THEN
32010
```

More next month
gi

For information, please call one of the people listed below !

President	Tracy Webber	661-5751
Vice Pres	John Hauser	458-0595
Newsletter	Jim Salmon	879-8119
Repairs	Walt Gremillion	495-2861
HACE BBS	24 Hours	458-9923

Please call before 9pm, and leave a message if necessary. If you can assist others, add your name here!

OCTOBER AGENDA

6:30 Equiptment set-up
Buy & Sell tables
Newsletter table

6:45 GRAPHIC DEMOS
Several "slide show" presentations, and other graphics demos will be running until the Formal meeting staRts at 7:15

7:15 GRAPHICS PRESENTATIONS
——Colourspace, presented by Tracy Webber
——Atari Artist, presented by John Hauser
——BBK Artist, presented by Jim Salmon

7:30 GRAPHICS UTILITIES
——Iconverter, presented by Jim
——Video Titler, presented by Jim
——PS to Newsroom Converter, presented by Tracy
——Technicolor Dream, presented by Jim

8:00 CLUB BUSINESS
——Officer Reports
——Election Nominations
——Membership Brochure Update
——Announcements

8:30 SALES & SPECIAL INTEREST GROUPS
We will break into groups interested in seeing how the graphics programs work, and give everyone a chance to try them. Also, members working on the brochure will get together.

NOTE: NO equiptment/software sales are permitted during the formal meeting from 7:15-8:30. Please arrive early to set up your tables!

We can use extra equiptment for the demos. Please bring whatever you can.

Also, there will be no aerobics classes this week, so come early!

EIGHT TIPS

George Iken
Houston Atari Computer Enthusiasts HACE 11/12-88

This month's tips for the eight bit are based on two scrolling demos by Charles Zubieta found in the October 1983 Atari Outpost column of Creative Computing. I have assumed the chance of every HACE member having immediate access to that article is slim, so the demos are reprinted here (with only a little modification). I am not much of a machine language programmer (you may insert NOT AT ALL, and be pretty much right), so the use of BASIC strings to control scrolling is a nice trick.

1. Speed and simplicity are hallmarks of graphics manipulation by means of strings. In both demos, lines 10 through 80 are identical. The idea is that the string A$ will be substituted into screen memory at will. Those 7 lines are the code that locates screen memory locates the string A$, and changes the pointers in the variable table so that the string is now part of screen memory. The other lines of code define the string and define how the string will interact with screen memory.

2. Possible uses of string manipulation include horizontal scrolling (as in the first demo), vertical scrolling (as in the second demo), mixed scrolling, etc. Possible applications listed in the article (no details though) include fast screen animation, high resolution drawings displayed almost instanteously, fast text handling on the screen, multiscreen playfields, etc.

3. Both demonstrations use joysticks to control scrolling. Joysticks can provide any of nine values to the computer through the joystick registers (STICK(0) for PORT 1 and STICK(1) for PORT 2 on XL and XE machines, and on the older 800 and 400 models, you also have STICK(2) for PORT 3 and STICK(3) for PORT 4). The nine positional values for a joystick 15 (dead center), 14 (push up), 13 (push down), 11 (push left), 7 (push right), 6 (up and right), 5 (down and right), 9 (down and left), and 10 (up and left). The demos don't use the joystick trigger, but as an example, the value of STRIG(0) would be zero WHEN THE TRIGGER OF STICK(0) IS PRESSED.

4. As stated in the demos, the internal character set is used in making up the string. This presents a few problems in that the screen takes that internal character value and uses it as though it is an ATASCII value. Life is easier if you have access to both ATASCII and Internal Character charts, but if you only have one, the following formulas can be used for conversion:

For ATASCII codes 0 through 31, add 64 to get the INTERNAL code.

FOR ATASCII codes 32 through 95, subtract 32 to get the INTERNAL code.

FOR ATASCII codes 96 through 127, the INTERNAL code is the same as the ATASCII code. (That is one reason why the demos show mostly lowercase letters in the strings, because they are in the range of 97 to 122 ATASCII, and do not need conversion with the INTERNAL code.)

5. Ok, I'm feeling sorry for you, here's the upper case and numeric conversion between ATASCII (0 through 95) and INTERNAL (Graphics characters (ATASCII 0 through 31) can't be reliably printed, and are defined only by keystroke, sorry):

ATASCII/KEYSTROKE/INTERNAL

0/ctrl ,/64 * 1/ctrl A/65 * 2/ctrl B/66
3/ctrl C/67 * 4/ctrl D/68 * 5/ctrl E/69
6/ctrl F/70 * 7/ctrl G/71 * 8/ctrl H/72
9/ctrl I/73 * 10/ctrl J/74 * 11/ctrl K/75
12/ctrl L/76 * 13/ctrl M/77 * 14/ctrl N/78
15/ctrl O/79 * 16/ctrl P/80 * 17/ctrl Q/81
18/ctrl R/82 * 19/ctrl S/83 * 20/ctrl T/84
21/ctrl U/85 * 22/ctrl V/86 * 23/ctrl W/87
24/ctrl X/88 * 25/ctrl Y/89 * 26 ctrl Z/90

27 through 32 are generally not used in strings (esc and arrow keys)

32/sp/0 * 33/!/1 * 34/"/2 * 35/#/3
36/$/4 * 37/%/5 * 38/&/6 * 39/'/7
40/(/8 * 41/)/9 * 42/*/10 * 43/+/11
44/,/12 * 45/-/13 * 46/./14 * 47///15
48/0/16 * 49/1/17 * 50/2/18 * 51/3/19
52/4/20 * 53/5/21 * 54/6/22 * 55/7/23
56/8/24 * 57/9/25 * 58/:/26 * 59/;/27
60/</28 * 61/=/29 * 62/>/30 * 63/?/31
64/@/32 * 65/a/33 * 66/B/34 * 67/C/35
68/D/36 * 69/E/37 * 70/F/38 * 71/G/39
72/H/40 * 73/I/41 * 74/J/42 * 75/k/43
76/L/44 * 77/M/45 * 78/N/46 * 79/O/47
80/P/48 * 81/Q/49 * 82/R/50 * 83/S/51
84/T/52 * 85/U/53 * 86/V/54 * 87/W/55
88/X/56 * 89/Y/57 * 90/Z/58 * 91/[/59
92/\/60 * 93/]/61 * 94/^/62 * 95/_/63

6. You may have noticed (or will, if you try the demos) that the space bar ATASCII code shares that value with the internal code for the CTRL COMMA key combination. In other words, if you type the demos in, you need to type CTRL COMMA (only in the STRING lines) wherever a space is shown. In fact, there's a whole lot of key combinations with the CONTROL key. With all that activity, you can accidentally hit the CONTROL and CAPS key together. Well folks, that locks you into graphics mode. Here's a tip back from the September 88 column: Press the SHIFT and CAPS key simultaneously to get OUT of a graphics mode lock.

7. HORIZONTAL SCROLLING DEMO (remember to enter a CONTROL COMMA instead of a space in the lines that define MESSAGE$)

```
1 REM from Creative Computing 10/83, modified
by G.N. Iken for the Houston Atari Computer
Enthusiasts, 11/88

10 DIM A$(960)

20 VTABLE=PEEK(134)+256*PEEK(135)

30 SCREENRAM=PEEK(88)+256*PEEK(89)

40 OFFSET=SCREENRAM-ADR(A$)

50 V3=INT(OFFSET/256)
```

```
60 V2=OFFSET-256*V3

70 POKE VTABLE+2,V2

80 POKE VTABLE+3,V3

100 SETCOLOR 2,0,0:SETCOLOR 1,0,15:SETCOLOR 4,8,4

120 DIM MESSAGE$(1001)

130 MESSAGE$=" the message now being scrolled is written lower case because the internal character set is"

140 MESSAGE$(LEN(MESSAGE$)+1)="  directly compatible with atascii only in the lower case however "

150 MESSAGE$(LEN(MESSAGE$)+1)="upper case can be scrolled if the atascii to internal conversion is made  "

160 MESSAGE$(LEN(MESSAGE$)+1)="for instance this message is brought to you by the "

170 MESSAGE$(LEN(MESSAGE$)+1)= "(/534/. !4!2) #/-054%2 %.4(53)!343      your !4!2) support group "

180 MESSAGE$(LEN(MESSAGE$)+1)= "meeting the 4th wednesday of each month at 1567 #olonial (ouse apartments "

190 MESSAGE$(LEN(MESSAGE$)+1)="call 713 493 2310 for registration information  "

200 ? CHR$(125):POKE 752,1

210 POSITION 4,1:? "Houston A.C.E. November, 1988":POSITION 1,23:? "C Zubieta, Creative Computing, 10/83"

212 POSITION 0,19:? "  CONTROL SCROLL SPEED WITH JOYSTICK  ":POSITION 11,20:? " RIGHT OR LEFT "

220 PI=1:NI=LEN(MESSAGE$)-40:SLOW=15

230 A$(121)=MESSAGE$(PI,PI+39)

236 A$(361)=MESSAGE$(NI,NI+39)

242 A$(601)=MESSAGE$(PI,PI+39)

250 FOR SPEED=1 TO SLOW:NEXT SPEED

260 IF STICK(0)=15 THEN ST=0

270 IF STICK(0)=11 THEN ST=5

280 IF STICK(0)=7 THEN ST=-5

290 SLOW=SLOW+ST

292 IF SLOW<1 THEN SLOW=1

294 IF SLOW>125 THEN SLOW=125

300 TRAP 220:PI=PI+1:NI=NI-1

500 GOTO 230
```

8. VERTICAL SCROLLING DEMO (remember to enter a CONTROL COMMA instead of a space in the lines that define M$)

```
1 REM from Creative Computing 10/83, modified by G.N. Iken for the Houston Atari Computer Enthusiasts, 11/88

10 DIM A$(960)

20 VTABLE=PEEK(134)+256*PEEK(135)

30 SCREENRAM=PEEK(88)+256*PEEK(89)

40 OFFSET=SCREENRAM-ADR(A$)

50 V3=INT(OFFSET/256)

60 V2=OFFSET-256*V3

70 POKE VTABLE+2,V2

80 POKE VTABLE+3,V3

90 DIM M$(2900)

100 SETCOLOR 4,8,4:SETCOLOR 2,0,0:SETCOLOR 1,0,15

102 ? CHR$(125):POKE 752,1

105 ? "MOVE JOYSTICK UP OR DOWN TO SCROLL":POSITION 2,22:? "C. Zubieta, Creative Computing 10/83"

106                 " M$(LEN(M$)+1)="

107 M$(LEN(M$)+1)="make each line forty characters long  "

108                 " M$(LEN(M$)+1)="

109                 " M$(LEN(M$)+1)="

110 M$(LEN(M$)+1)="(/534/. !4!2) #/-054%2 %.4(53)!343  "

120                 " M$(LEN(M$)+1)="

130 M$(LEN(M$)+1)="the internal character set is used so  "

140 M$(LEN(M$)+1)="only atascii codes ninetysix through  "

150 M$(LEN(M$)+1)="atascii codes onehundredtwentyseven are "

160 M$(LEN(M$)+1)="directly translated "

170                 " M$(LEN(M$)+1)="

180 M$(LEN(M$)+1)="this means the lowercase letters and a  "

190 M$(LEN(M$)+1)="few symbols will show up on your display"

200 M$(LEN(M$)+1)="just like you typed them in basic  "

210                 " M$(LEN(M$)+1)="

220 M$(LEN(M$)+1)="the internal character set can be  "

230 M$(LEN(M$)+1)="related to atascii by the following  "
```

```
240                    M$(LEN(M$)+1)="formula
                "

250                                M$(LEN(M$)+1)="
                "

260  M$(LEN(M$)+1)="atasci  code  thirtytwo
through ninetyfive"

270 M$(LEN(M$)+1)="subtract thirtytwo to get
internal code "

280                                M$(LEN(M$)+1)="
                "

290 M$(LEN(M$)+1)="atascii code zero through
thirtyone      "

300  M$(LEN(M$)+1)="add  sixtyfour  to  get
internal code      "

310                                M$(LEN(M$)+1)="
                "

320 M$(LEN(M$)+1)="atascii code ninetysix to
onetwentyseven"

330 M$(LEN(M$)+1)="is the same as internal
code               "

340                                M$(LEN(M$)+1)="
                "

350 M$(LEN(M$)+1)="what this means is that to
print an       "

360  M$(LEN(M$)+1)="atascii  character  to
screen                "

370 M$(LEN(M$)+1)="the basic listing has to
be converted    "

380 M$(LEN(M$)+1)="so that the internal code
of the         "

390  M$(LEN(M$)+1)="character  you  want  to
display is the     "

400 M$(LEN(M$)+1)="same as the atasci code of
the character"

410 M$(LEN(M$)+1)="used in the basic listing
               "

420                                M$(LEN(M$)+1)="
                "

430 M$(LEN(M$)+1)="for example lets look at
the spacebar    "

440 M$(LEN(M$)+1)="which is atascii thirtytwo
                "

450  M$(LEN(M$)+1)="so  minus  thirtytwo  for
internal code      "

460  M$(LEN(M$)+1)="you  need  to  enter  the
character which    "

470  M$(LEN(M$)+1)="has an atascii code of
zero and that is "

480 M$(LEN(M$)+1)="the control and comma keys
pressed        "

490 M$(LEN(M$)+1)="simultaneously          ie a
heart symbol "

500                                M$(LEN(M$)+1)="
                "

510 M$(LEN(M$)+1)="similarly to get a capital
t which is     "

520  M$(LEN(M$)+1)="atasci  eightyfour  minus
thirtytwo for an"

530 M$(LEN(M$)+1)="internal code of fiftytwo
                "

540  M$(LEN(M$)+1)="enter  whichever  chracter
has an atasci    "

550    M$(LEN(M$)+1)="value    of    fiftytwo
ie the number "

560 M$(LEN(M$)+1)="four                so enter a
four in basic "

570  M$(LEN(M$)+1)="and  see  a  capital  t  on
screen
                "

580                                M$(LEN(M$)+1)="
                "

590                                M$(LEN(M$)+1)="
                "

600 M$(LEN(M$)+1)="an easy fast scroll using
Strings        "

610 M$(LEN(M$)+1)=" but remember the internal
character set"

620                                M$(LEN(M$)+1)="
                "

630                                M$(LEN(M$)+1)="
                "

1000 S=1:ST=40

1010 A$(81)=M$(S,S+799)

1015 TRAP 2000

1020 IF STICK(0)=14 THEN ST=-40

1030 IF STICK(0)=13 THEN ST=40

1035 IF STICK(0)=15 THEN ST=0

1040 D=S:S=S+ST

1050 GOTO 1010

2000 S=D:FOR D=1 TO 40:NEXT D:GOTO 1010
```

More next month
gi

(13)

GENLOCK

Professional Level Desktop Video

Review By Bill Moes

The Mighty Word

Genlock. The word has a certain magic for graphics enthusiasts, a magic that offers dreams of Disney-like creations, combining the imaginative possibilities of computer animation along with the reality of standard video. Genlock. A word that makes all other computer terms sound hollow and empty. True desktop video! Dynamic, creative, exciting ... and on the ST!

In understandable terms, a genlock allows you to superimpose computer graphics on standard video images. For example: you've taken your camcorder on a family vacation and have come back with stunning video shots (sounds about right, doesn't it?). A genlock on your ST will allow you to add computer-generated titles, special effects, or animation to those video frames.

Titling & Other Effects

Genlocks could also be used at public access cable TV channels for program titling and other effects. And other professionals who use video (weddings, etc.) could find a genlock of great value.

John Russell, who spent the past 1 1/2 years developing a genlock for the ST, sounded understandably enthusiastic about his device when he recently discussed it with *Current Notes*.

Citing the Cyber animation software from Antic Publishing, along with the other animation titles available for the computer, Russell stated that the ST has "the finest animation programs going now" and added that his genlock clearly "puts the ST into the desktop video world."

In addition to the excellent graphics and animation software already available to the ST community, Russell mentioned that software developers were working on programs to be used specifically with his genlock device.

FCC Approval

Russell's genlock is a plug-in board to cost approximately $400. It will be initially available only for the Mega ST. The first units will be bundled (Mega ST and installed genlock board) and will be sold by JRI (John Russell Innovations). It's all waiting for FCC approval (a familiar phrase to all computer users). The package should be available, though, sometime this summer. Later, Russell plans to have the board available separately for user installation, a solderless process.

Near the end of the year, he hopes to have a genlock available for most other ST's (520 ST-FM and 1040 ST), although he cautioned that users who have installed certain memory upgrade boards may have a problem in also installing the genlock board due to space considerations. If you have an ST with a memory upgrade, you'll have to check out your specific situation. Some memory upgrade boards could be a problem; some will not.

Russell, whose background includes research and development with electronic test equipment, mentioned that the very early 520 ST's would probably not be able to work with the genlock because of the way those computers were initially assembled and the assembly variations since then.

In mildly technical terms, a genlock allows you to have video-in as your external source signal. The genlock will then reconfigure the machine timing to produce RS170 output. Timing is used by a computer to determine when things happen and at what level.

A genlock, then, is used to lock the computer to your external video source so you can superimpose computer-generated images on those external video images. You'll then be able to view the composite image on your RGB monitor and tape them with a second VCR.

The genlock, incidentally, will work in either the ST's low or medium resolution. It will be totally transparent to the ST when it's not being used and, therefore, should not interfere with normal software programs. Russell added that a full list of the genlock's features can not be made until it receives final FCC approval.

So the promise of true desktop video on the ST should be soon met. We've watched as other computers, those without the powerful and sophisticated family of graphics software found on the ST, have had genlocks available.

And now it's the ST's turn. And ours.

[John Russell Innovations (JRI), P. O. Box 5277, Pittsburg, CA 94565 (415) 458-9577]

NEAFT CPAUG

(14)

GENERAL INTEREST

EX-PROGRAMMER GUILTY IN COMPUTER "VIRUS" CASE

From the Arizona Republic, September 21, 1988

The Associated Press: Fort Worth, Texas -

A former programmer has been convicted of planting a computer "virus" in his employer's system that wiped out 168,000 records and was activated like a time bomb, doing its damage two days after he was fired. Tarrant County Assistant District Attorney Davis McCown said he believes he is the first prosecutor in the country to convict someone for destroying computer records using a virus.

"We've had people stealing through computers, but not this type of case," McCown said. "The basis for this offense is deletion." "It's very rare that the people who spread the viruses are caught," said John McAfee, chairman of the Computer Virus Industry Association in Santa Clara, Calif., which helps educate the public about viruses and find ways to fight them. "This is absolutely the first time" for a conviction, he said. "In the past, prosecutors have stayed away from this kind of case because they're too hard to prove," McCown said Tuesday. "They have also been reluctant because the victim doesn't want to let anyone know there has been a breach of security."

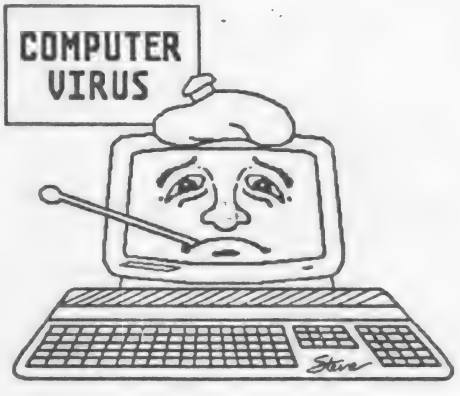
Donald Gene Burleson, 40, was convicted Monday of charges of harmful access to a computer, a third-degree felony that carries up to 10 years in prison and up to $5,000 in fines. A key to the case was the fact that state District Judge John Bradshaw allowed the computer program that deleted the files to be introduced as evidence, McCown said, adding that it would have been difficult to get a conviction otherwise.

The District Court jury deliberated six hours before bringing back the first conviction under the state's 3-year-old computer- sabotoge law. Burleson planted the virus in revenge for his firing from an insurance company, McCown said. Jurors were told during a technical and sometimes-complicated three-week trial that Burleson planted the rogue program in the computer system used to store records at USPA and IRA Co., a Fort Worth based insurance and brokerage firm.

A virus is a computer program, often hidden in apparently normal computer software, that instructs the computer to change or destroy information at a given time or after a certain sequence of commands. The virus, McCown said, was activated Sept. 21, 1985, two days after Burleson had been fired as a computer programmer because of personality conflicts with other employees. "There were a series of programs built into the system as early as Labor Day (1985)," McCown said. "Once he got fired, those programs went off". The virus was discovered two days later, after it had eliminated 168,000 payroll records, holding up company paychecks for more than a month. The virus could have caused hundreds of dollars in damage to the system had it continued, McCown said.

The defense argued during the trial that Burleson was set up by someone using his terminal and code. Burleson's attorneys tried to prove he was vacationing in another part of the state with his son on the dates in early September when the rogue programs were entered into the system. But prosecutors presented records showing that Burleson was at work and his son at school on those dates. Defense lawyer Jack Beech maintained Burleson is innocent but said his client might not have enough money to appeal. Bradshaw on Tuesday ordered a pre-sentence investigation and set sentencing for Oct. 21.

Typed from The Arizona Republic, by Les Deman Sept. 22, 1988



8 Bits at a time 11-88

A Parallel Disk Interface for your XL/XE

Once you have taken the plunge and purchased a computer, you soon learn that the computer itself requires a number of additional devices in order to be really useful. Of prime importance is the disk drive. The computer contains 64K of Random Access Memory that is wiped clean when you turn on the power. Therefore, after power-up, you need to enter whatever program you want to run into all this memory. The XEGS uses cartridges that contain the programs and thereby allows you to play games without a drive (which will work on most any 8-bit, by the way). For almost any other activity on your system, you must load a program from your drive (called booting the system).

The path that the data must take from the drive to the system is called the Serial Input Output interface, SIO. The input/output refers to the fact that data travels in and out of the system on this buss. Serial refers to the method used to send each group of characters, called bytes. A character is a group of 8 bits that define a symbol or command to the system. And, a bit is just a two position flag that signifies *on* or *off*, *one* or *zero* - a two state indicator (*B*inary Digi*t*). In order to transmit these bits from one place in the computer to another, we need a conductor, or a path. Each conductor (a piece of wire in our case) can only send one bit at any moment in time, so if we want to send one byte (eight bits), we will need eight wires (referred to as a *parallel* interface). Because of the cost and complexity of producing multi-conductor cables and connectors, a simpler method was devised to transmit data outside the computer, *serial* transmission. Instead of sending all eight bits at once, the byte is broken down into individual bits which are sent in a specific order at regular intervals. The destination assembles this series of bits back into a byte for use by the system or device. This operation only requires one wire, not eight and is the preferred method of transmission outside the computer. The problem, of course, is that the serial method takes about eight times longer to transmit a character than the parallel scheme. Not such a big deal for a printer or modem, but significant on a disk drive.

The Atari SIO interface can only transmit about 1000 characters per second under normal conditions, which means that a 22,000 byte file will take 22 seconds to load or save. A 45,000 byte dictionary for a spelling checker will take 45 seconds. A 192K ramdisk will take 192 seconds. A ramdisk? 192 seconds? I could go eat lunch while that loaded! Even these times assume that the data is sequenced properly on the diskette. If the drive has to search around on the disk for the data, the data transfer time can be much, much higher. This is why a ramdisk is so useful - when you have a ramdisk, data is instantly available. But, what can be done about the 192 seconds it takes to load the ramdisk? If the ramdisk is going to save you time, we can't be taking over three minutes to change a couple of "diskettes", right? You bet.

The solution is to load the data into the ramdisk using a parallel interface, eight bits at a time - eight times faster than the SIO. Then, it would take 24 seconds to load two standard Atari diskettes into a ramdisk. This, I can live with. You could also use the parallel interface directly in a program (to save a file someplace not likely to be bothered by a power surge, for example). While some of us don't mind doing a little soldering inside our computers, I think a number of users would rather not open up the nice computer and turn it into a box of rocks, so this interface will plug into the PBI connector on the back of your XL or XE. You don't need to open up your computer. You don't need to rip up your XF551, either. Or your 1050..... You did get a couple of those neat-o Radio Shack $99 drives, didn't you? This interface will run up to four of the R/S drives, or four SF314s, or SF354s, or XF551s, or..... 3.5 or newer style 5.25 drives can even be mixed together.

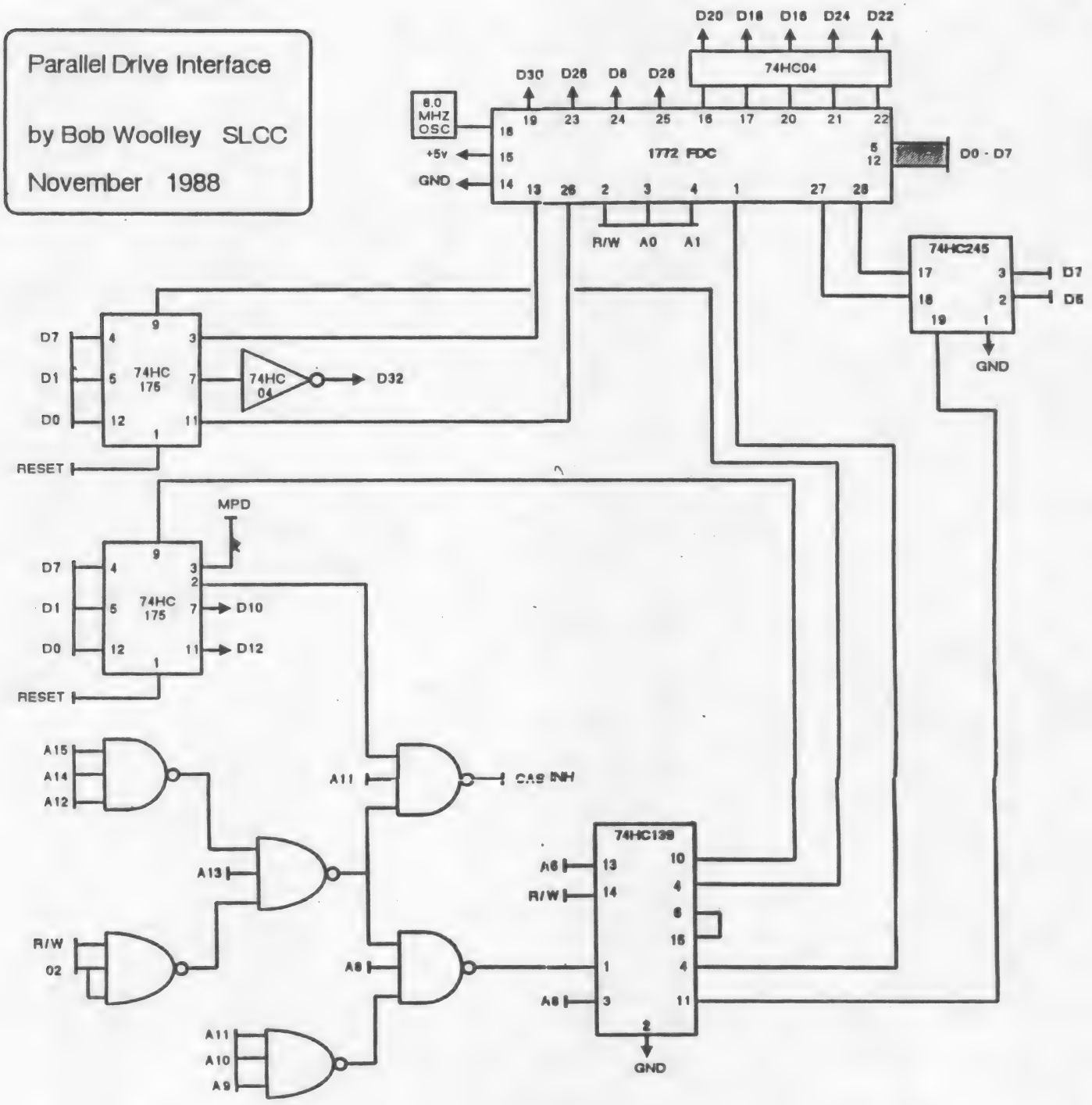
The schematic for the interface is presented this month. The software required to run the system will hopefully be finished by next month, although a number of options may take longer than that to write. As it presently stands, the drive will only read single density, 128 byte sectors. Double density sectors do not allow enough time between bytes for the system to refresh memory and handle the non-maskable interrupt without losing data. I can read most of the 512 byte ST/IBM sector before getting lost, but that is obviously not good enough.This will limit normal operation to 360K of data per drive instead of 720K and about 8,000 bytes of data per second. Not too bad, considering. Once the DD problem is solved, each drive will hold 720K and transfer over 10,000 bytes per second.

I used a Radio Shack drive #25-1061 and a Coleco power supply also available from R/S as #277-1022. You have to remove the existing cables from the drive and make up a 34 pin cable for the interface, as well as splice the power connector from the power supply. If we are lucky enough to get one of the Atari specific suppliers to produce these things, a complete kit may be available for the project. That would be a lot faster than waiting for me to make a PC board for this thing. Until then, those of you that thrive on flux vapors can get busy.

Until next month!

Bob Woolley SLCC

75126,3446



A0 | REPRESENTS A CONNECTION TO THE PBI 50 PIN BUSS

→ D24 REPRESENTS A CONNECTION TO THE 34 PIN DRIVE CABLE

Programming the ST
with
Personal Pascal
Part 2

10-88

Paul Machiaverna - JACG

Last month I showed a couple of very simple TOS programs for you to try typing in, compiling, and running with DSS Personal Pascal (PP). This month I will cover writing some simple GEM programs. The simplest GEM function to use is the Alert Box. Alert boxes are what you see when a program informs you to do something, or something is wrong. You can see an Alert box on you screen by taking the disk out of your disk drive and opening the drive icon (double-click on the drive icon). The ST will try to read a disk in the drive a few times then display an Alert box telling you that an error has occurred. This is the type of GEM function I will cover here this month.

One comment I must make before we begin is concerning remarks placed in Pascal programs. In last month's article it was not clear as to what was a remark and what was the actual program. Unfortunately, the required left and right curly brace characters used to tell Pascal that the lines are remarks did not appear because the editor of this news letter is using the 8-bit Atari computer which does not support these characters as text. So, from now on I will use the older notation which is (*, to mark the beginning of a remark, and *), to mark the end of a remark. For example, (* This is a Pascal remark *). However, I suggest that you replace all (* with the left curly brace, and *) with the right curly brace at all times when you copy the programs. In some cases the right and left curly braces are used to give instructions to the compiler. In this case the (* *) pair will not work.

Let's start by covering a few important points about writing GEM programs. PP has to be told that we are writing a program to be compiled for GEM. This will make the compiler use the proper routines for creating a GEM program, and the linker will use the proper libraries for producing an executable file with the PRG file extender. So, the first thing to do is pull down the options menu from the PP manager and select compile and link for GEM. Next we have to include a library file in our source code to inform the compiler where to find the routines to support the GEM calls in our program. Finally, our program has to ask the ST if it can use the GEM environment when we try to run our executable file. The reason for this is that a GEM application needs to be initialized, meaning that our program will be using the GEM functions. Now, let's consider our first, simple GEM program.

As I stated before, Alert boxes are the simplest of all GEM functions to use. However, their simplicity of use also includes their simplicity in what they can do. Alert boxes are useful for simple halting of a program to inform the user of a error, to instruct them to do something, or to get simple predetermined input. Alert boxes can contain an icon to enhance their appearance and convey the nature of it's reason for being on the screen. You can choose one of three icons or none at all. The three possible icons are the exclamation point, the question mark, and the stop sign. No icon in an alert box is good for program titles or conveying simple information to the user. The exclamation point is good for grabbing the users attention to important information, like if you wanted to say 'Clicking on OKAY will format your disk!' That's an example of a operation which would require the user to make a very important choice. The question mark is good for telling the user that some unknown problem has occurred or that the program does not understand some input. The stop sign is good for instructing the user to stop and doing something like 'Be sure that your printer is online.' The impact of icons on the user are tremendous. Pictures convey information much quicker and vivid to the user than is possible with words. For this reason, it is a good idea for you to think about how you want an Alert box to appear.

Take a careful look at the program below which will show all the possible Alert boxes. Incidentally, only one Alert box can be displayed on the screen at once. Now type in the program being very careful to replace all (* *) shown with the respective curly braces. Where you see (*$I GEMSUBS.PAS*) in the program you must replace the (* *) or the program will not compile. After you compile and run the program to see it's action we will discuss each line in detail to understand what it is doing.

```
(* Demonstration of the four types of the GEM Alert Boxes.  They
are:
  1) No Icon displayed
  2) Exclamation point in an inverted triangle
  3) Stop Sign
  4) Question mark in an inverted triangle

     Written by Paul Machiaverna - JACG    September 12, 1988 *)

     Program Four_Alert_Boxes ;

     (*$I GEMSUBS.PAS*)  (* Include the GEM routines *)

     Var
Choice : Short_Integer ;  (* value returned by alert box *)

     Begin
If Init_Gem >= 0 then  (* request the use of GEM *)
Begin
Choice := Do_Alert('[0][  Alert Box! with No Icon  ][  Next
]',1) ;
Choice := Do_Alert('[1][  Alert Box!  with an! Exclamation Point
][  Next  ]',1) ;
Choice := Do_Alert('[2][ Alert Box! with a! Question MArk!][  Next
]',1) ;
Choice := Do_Alert('[3][  Alert Box  ! with a! Stop  Sign!][  End
]',1) ;
Exit_Gem  (* we are done using the GEM routines *)
End
End.
```

Congratulations! You have just completed a GEM program. If you have problems with compiling and linking the program the first thing to check is that you used the curly braces instead of the (* *) on the line which includes the GEM routines, as discussed above. If you still have problems, make sure that you chose compile and link for GEM from the PP manager pull down menu. Now let's take the source code apart and understand each line.

The program starts off with a comment which tells what the program does. I can not emphasize the importance of commenting your source code enough. Spend a little time including comments when you write a program instead of spending a lot of time later trying to figure out what your program does. The next line is the standard required Program statement found in every Pascal program. The most important line of this program is the {$I GEMSUBS.PAS} directive. It tells the compiler to include the GEM subroutines required to write a GEM program. The source code, or any other GEM program, will not compile with out this directive. GEMSUBS.PAS is a file which is written in Pascal syntax and includes all the calls necessary for the linker to properly make an executable GEM program. The file is included on the PP disks. The integer value called 'Choice' is used to hold the value returned by the Alert box. This value is the choice of the user and is made by clicking on a box with mouse or hitting return on the keyboard to choose the default choice.

Now we have to request the use of GEM by using the Init_Gem function. From how I understand it, GEM can only handle a certain amount of work at a time. You have to ask if GEM can initialize an application before you use it, such as displaying an Alert box. The value returned must be greater than or equal to 0 if we are to run our GEM application. A negative value basically means that GEM is telling us that our application cannot be initialized for use. The successful return value (>= 0) of Init_Gem is only useful when writing Desk Accessories. Thus, this return value is not needed for any other use in our program.

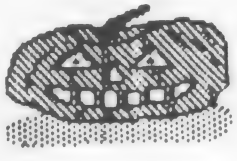
The actual call to the Alert box to make it appear on the screen is done with the Do_Alert function. Do_Alert() is called a function because it is called by assigning it's return value to a variable. Therefore, we cannot call up the Alert box by simply writing Do_Alert(). We must write it as Choice := Do_Alert(). Choice was declared as a Short_Integer because this is the type of value returned by calling the function. The () parentheses denote that there are parameters required by the function. These parameters tell GEM how the Alert box is to appear and how to handle the button(s) of the box. The buttons of an Alert box are the rectangles to which the user moves the mouse pointer and clicks on to exit the box and return a value to the program. The parameters of the Alert box are 1) a string which contains information about the type of icon to display, the text to appear in the box, and the names of the buttons contained in the box, 2) a number which tells GEM which button to consider the default, if any. The default condition of a button is when the user can invoke the highlighted button by either clicking on it or hitting the return key. You may choose to have any button from 1 to 3 as the default or no default by coding a 0 for this part of the string. Note that each part of the string in contained by left and right brackets, [and]. The vertical lines in the text part of the string denotes a new line for each vertical line shown. This means that you can have more than one line of text in an Alert box.

The program continues by displaying each type of Alert box possible. It will show one box, wait for the user to click on a button or hit return, and show the next type, until all four have been shown. The last function we encounter in this program is Exit_Gem. This tells the computer that we are done using GEM. You should not call this function if the call to Init_Gem returned an unsuccessful initialization value of less than zero. Unpredictable results can occur if this rule is violated.

The use of Alert boxes are simple, but can be frustrating. They are simple because you only have to write very little code to create and display them (one line!). They are frustrating because you don't have a lot of control over their appearance and there are some cautions to abide by to avoid a system crash. The PP owner's manual gives you a list of cautions when programs which use Alert boxes. As for what to consider, I'll point out a few things in the use of Alert boxes. The only control you have over the size of the Alert box is dictated by the string which you enter as the parameter to the function. Getting text to be centered in the box requires some experimenting at first. No patterns can be used to enhance the appearance of the box. Alert boxes are always centered on the screen. Your program will stop dead waiting for the user to click on one of the buttons or hit return if enabled with the default condition parameter. Overall, remember that Alert boxes are intended to grab the user's attention (hence, 'Alert') to something important when running a program.

Experiment with creating all different types of Alert boxes by writing simple programs to test them. Learn how to properly center the text in an Alert box. Know how to use the right type of icon to convey the nature of the appearance of the Alert box. Alert boxes are ideal for learning how to write programs for GEM. Eventhough there are limitations to there use, Alert boxes are an integral part of many GEM programs and are very useful because of their attention grabbing quality.

I hope you found this discussion of Alert boxes interesting. Above all, you now know some basic considerations for writing GEM programs. The GEM user interface is very powerful. When writing any GEM program you should think in terms of the user and do whatever is necessary to make your program easy to use. Next month I will break from GEM programming and discuss how to write TOS programs which tap the potential of the screen formatting routines available in PP which allow for easy formatting of text which are not available in standard Pascal. Finally, I will make the source codes of the programs shown in these articles available on the JACS BBS for those who don't like to type in programs. However, I will leave the compiling up to you for the experience. See you next month.



ATARI'S SMALL MIRACLES

by Joseph Russek

(19)

STAR

A series of colored lines emanating from the center of a black screen combine to make quite an impressive star. Thanks to the Eugene, Oregon, Atari Computer Enthusiasts for this small gem.

```
0 REM EUGENE ATARI COMPUTER ENTH-
1 REM USIASTS EXCHANGE LIBRARY
2 REM **** A STAR IS BORN ****
3 REM BY JON LOVELESS
4 REM C/O MATT LOVELESS
5 REM 18623 PLUMOSA ST.
6 REM FOUNTAIN VLY., CA 92708
7 POKE 77,0
10 GRAPHICS 7+16
20 FOR A=1 TO 48:COLOR A
40 SETCOLOR 4,INT(16*RND(1)),1
50 FOR X=0 TO 157:Y=INT(96*RND(1))
70 PLOT 79,49:DRAWTO X,Y:NEXT X
80 DRAWTO X,Y
100 NEXT A
```

ROLL

Slowly one-by-one four blue cylinders are formed. When they are completed, they begin to rotate, creating a cartoon-like three-dimensionality.

```
5 DEG
10 GRAPHICS 10
15 FOR I=0 TO 7:POKE 705+I,128+2:NEXT I
17 POKE 705,136
20 FOR ANG=180 TO 360+180 STEP 6
30 X=8+8*COS(ANG)
40 Y=16+8*SIN(ANG)
50 COLOR (ANG-180)/45+1:PLOT X,Y
60 DRAWTO X,50+Y
70 COLOR 0:PLOT X,Y
90 NEXT ANG
120 FOR ANG=180 TO 360+180 STEP 6
130 X=26+8*COS(ANG)
140 Y=16+8*SIN(ANG)
150 COLOR 9-(ANG-180)/45:PLOT X,Y
160 DRAWTO X,50+Y
170 COLOR 0:PLOT X,Y
190 NEXT ANG
220 FOR ANG=180 TO 360+180 STEP 6
230 X=44+8*COS(ANG)
240 Y=16+8*SIN(ANG)
250 COLOR (ANG-180)/45+1:PLOT X,Y
260 DRAWTO X,50+Y
270 COLOR 0:PLOT X,Y
290 NEXT ANG
320 FOR ANG=180 TO 360+180 STEP 6
330 X=62+8*COS(ANG)
340 Y=16+8*SIN(ANG)
350 COLOR 9-(ANG-180)/45:PLOT X,Y
360 DRAWTO X,50+Y
370 COLOR 0:PLOT X,Y
390 NEXT ANG
410 GO TO 500
420 FOR ANG=180 TO 360+180 STEP 6
430 X=50+8*COS(ANG)
440 Y=16+8*SIN(ANG)
450 COLOR (ANG-180)/45+1:PLOT X,Y
460 DRAWTO X,50+Y
470 COLOR 0:PLOT X,Y
490 NEXT ANG
500 A=PEEK(705)
510 FOR I=705 TO 711
520 POKE I,PEEK(I+1)
530 NEXT I
540 POKE 712,A
550 GO TO 500
```

RAINBOW GTIA

This GTIA demo begins with the formation of a rectangular-colored spectrum. When each bar is completed, shifting of colors takes place, and a rainbow effect is achieved.

```
100 REM GTIA TEST
115 GRAPHICS 10:FOR Z=704 TO 712:READ R:
POKE Z,R:NEXT Z
116 DATA 0,26,42,58,74,90,106,122,138,154
130 FOR X=1 TO 8:COLOR X:POKE 765,X
140 PLOT X*4+5,0:DRAWTO X*4+5,159:PLOT X
*4+1,159:POSITION X*4+1,0:XIO 18,#6,0,0,"S:"
150 NEXT X
230 FOR X=8 TO 15:COLOR 16-X:POKE 765,16-X
240 PLOT X*4+5,0:DRAWTO X*4+5,159:PLOT X
*4+1,159:POSITION X*4+1,0:XIO 18,#6,0,0,"S:"
250 NEXT X
300 COLOR 0:PLOT 65,159:DRAWTO 0,159
400 FOR X=1 TO 8:Z=PEEK(704+X):Z=Z+16:IF
Z>255 THEN Z=26
420 POKE 704+X,Z:NEXT X:FOR Y=1 TO 5:NEX
T Y:GOTO 400
```

RAINBOW

A series of blue lines enter from the top, left-hand corner of the screen. When they are formed, they begin to vibrate and change colors. Thanks to the Salt Lake Atari Computer Enthusiasts for this interesting program.

```
10 REM RAINBOW
20 REM ACE OF SALT LAKE
30 GRAPHICS 15+16:COLOR 3
40 FOR X=0 TO 159 STEP 8
50 PLOT 0,0:DRAWTO X,191
60 NEXT X
70 FOR Y=191 TO 0 STEP -8
80 PLOT 0,0:DRAWTO 159,Y
90 NEXT Y
100 FOR I=0 TO 21:READ A:POKE 1536+I,A:NEXT I
110 A=USR(1536)
120 DATA 173,11,212,201,32,208,249,141,10,212,142
,24,208,232,232,208,246,142
130 DATA 24,208,240,232
140 END
```

The SpartaDos Cartridge: A Review

by MILE 1+1 11-88

Keith Joins

(Special thanks is given to Bill Aycock, another beta tester, for his work in editing and correcting this review and for adding his comments on some features I had failed to cover.)

By the time you read this the SpartaDos X cartridge, a long awaited release from ICD, should be on sale. The people who brought us the MIO, the US doubler, and the disk-based version of SpartaDos have now given us a DOS so fast and powerful that once you use it, you will never want to replace it with any other DOS currently available. After having used SDX for over two months while beta testing it, I can't understand how I ever survived without it.

The cartridge itself is similar in appearance to the R-Time 8. It is easily opened so that you are able to replace the socketed ROM. This is a fairly simple procedure. I have done it twice and if I can do it, anyone can. This will be the method used for any future upgrades--no soldering, thank you!

The cartridge allows for piggy-backing another cartridge, such as the R-Time 8, Action!, Basic XE, MAC-65, etc. I am using it with a MIO and the XE adapter. Tom Harker of ICD has said that they are looking into manufacturing some type of adapter for the XL to avoid the need to stack as many as three cartridges in the XL slot.

The cartridge is packed with 64K of code. There is no disk needed with SDX as all of the external command files are resident in the cartridge and load with amazing speed. The DOS is totally relocatable! The need for this will become apparent as you become familiar with SDX. It is this relocatability that has caused the long delay in completing SDX. Anyone who has ever tried to write an application program that is relocatable, let alone a system program, can appreciate the magnitude of this project.

Now let's take a look at some of the new features of SpartaDos X. Something that you have to learn at once is that there is now a new set of identifiers. The first one you learn is CAR:. This is the cartridge itself. We can issue the command DIR CAR: and get a directory of the cartridge as follows:

(21)

```
Volume: Cart 4.0
Directory: MAIN

SPARTA   SYS   7072  8-24-88  4:01p
MENU     COM   6688  8-25-88 12:01p
COMMAND  COM   4723  8-25-88 12:03p
INDUS    SYS   2498  8-11-88  5:09p
ARC      COM   3289  8-24-88  5:26p
X        COM   1999  8-24-88  4:01p
ATARIDOS SYS   1015  8-24-88  4:02p
CAR      COM   1600  8-24-88  4:01p
CACHE    SYS   1224  8-23-88  9:53a
UNERASE  COM   1056  8-24-88  9:55a
TD       COM   1016  7-22-88 12:07p
RAMDISK  SYS   1040  8-19-88  4:56p
SIO      SYS    970  8-16-88  5:34p
FIND     COM    831  8-25-88 12:53p
CLOCK    SYS    698  7-22-88 12:49p
JIFFY    SYS    614  7-25-88 10:48a
DUMP     COM    567  7-26-88 11:41a
KEY      COM    505  7-19-88  8:42a
CHTD     COM    372  7-26-88 11:47a
CHVOL    COM    298  7-12-88  9:10a
RPM      COM    388  8-25-88  1:50p
RS232    COM    147  7-07-88  3:59p
      0 FREE SECTORS
```

Remember that this is the beta version and the directory entries might be different on yours, but not much. A number of the .COM files will look familiar to seasoned SpartaDos users. These are the 'external' commands and they have the same functions in SDX as they did in the disk-based version.

The next new group of identifiers used is A: through I: (or 1: through 9:), which represent drives one through nine. D5K is now the identifier for the old D: and is always assumed. Therefore D: is no longer the default drive, but rather is drive four. So, you can reference drive one as 1:, D1:, D5K1:, or A: -- your choice. The default drive is now just a colon (:) since a single D preceeding a letter or a number is ignored.

Other identifiers are PRN: (the printer), CON: (the "console", or screen editor), and COM: (the RS-232 port).

This takes a little getting used to but becomes second nature after a bit. Any MS-DOS users will feel right at home. I learned CON: very quickly when attempting to create my first AUTOEXEC.BAT file (not STARTUP.BAT -- more MS-DOS compatability). I kept on trying to use the command:

```
COPY E: AUTOEXEC.BAT
```

I'll let you figure out what happened!

These new identifiers can be accessed from Basic by preceeding them with a 'D'. LOAD "D5:FILENAME" will load

filename from drive six. Of course you could still use LOAD "D6:FILENAME" also, so there's no problem with compatibility. If you take a look at the CAR: directory again, you will see some files with the extender ".SYS". These are handlers that you can choose from to configure your system the way you want. SDX will load several of these as a default configuration. SPARTA.SYS and SIO.SYS must always be loaded as they contain the actual SpartaDos disk driver and the SIO routines. The default configuration will also always load ATARIDOS.SYS and INDUS.SYS. ATARIDOS.SYS will allow you to access AtariDos type disks. INDUS.SYS will re-program any connected Indus or Happy-enhanced drives to enable high speed I/O. I have an Indus GT and this handler works great!

RAMDISK.SYS is the SDX ramdisk driver. You may install multiple ramdisks using any unused drive numbers in any size up to the total amount of available banked RAM. You specify the size of the ramdisk by entering the number of 16K banks to be used. For example:

RAMDISK.SYS 8 2

This will set up a ramdisk as drive eight using 2 16K banks of RAM. The ramdisk will be formatted automatically unless you had previously installed the ramdisk with the same number and size before performing a coldstart and without losing power to the RAM. This means that you can use the COLD command which re-boots the system and still retain the contents of your ramdisk. By the way, the ramdisk is formatted in double density.

CACHE.SYS, when installed, will use one 16K bank of extended RAM to add additional buffers to SPARTA.SYS to make floppies seem to run faster. This handler is not loaded as a default.

CLOCK.SYS and JIFFY.SYS are loaded dependent upon your system. If you have a R-Time 8 then CLOCK.SYS will load. Otherwise JIFFY.SYS will load as the default. One or the other must be used to give meaning to any Time/Date commands.

Each of the handlers except INDUS.SYS takes up memory. Therefore you don't want to use any that aren't needed. To do this you can set up a configuration file of your own. When SDX boots it will check drive one for a file called CONFIG.SYS. If the file is present, SDX will use that file for the list of handlers to load. If there is no CONFIG.SYS file, then the defaults will be used. Booting with OPTION held down will override any disk-based CONFIG.SYS file and will use the default configuration.

You can also choose what area of RAM SDX will use. The choices are OSRAM (RAM under the OS which is the default on a stock XL/XE), BANKED (expanded memory), or NONE (normal RAM which would usually be used only with a stock 800).

SpartaDos X now supports over 1,400 entries in a directory, a real blessing for hard disk users. There (22) are now two valid directory name separators; the usual ")" has been joined by the MD-DOS "\". Also, SDX recognizes the MS-DOS MKDIR, CHDIR, and RMDIR commands, which are equivalent to the old style CREDIR, CWD, and DELDIR. Note that the longest pathname allowed is still 63 characters.

There are several new commands available with SpartaDos X. The first of these is the ATR or Attribute command. This is really an enhancement of the old PROTECT and UNPROTECT commands which set or cleared a protection bit in the filename entry. The protection bit is now manipulated by the ATR command. You would use ATR +P FILE.EXT to protect a file, or ATR -P FILE.EXT to unprotect it. In addition, two new bits are available: the Archive bit and the Hidden bit. The Archive bit is cleared when a file is created or written to, and set when the file is backed up by a backup program, such as ICD's FlashBack!, which supports it. The Hidden bit allows you to hide a file or directory so that it can be loaded as a command only. Commands like DIR, COPY, and TYPE will not 'see' these files unless you specifically include the attribute with those commands.

The RUN command has been eliminated, the thinking being that it was never used except to re-boot the system (with RUN E477). To take the place of this, the COLD command has been added. COLD by itself causes a reboot. There are also two options, C and N, which disable SDX and allow you to boot up a disk-based DOS such as Atari DOS or an older version of SpartaDos. COLD /C will re-boot the system and leave an attached cartridge enabled, and COLD /N will disable all cartridges and reboot the system.

Users of SpartaDos will remember that you used the CAR command to return to the built-in Basic on your XE/XL or to the cartridge that you had installed. Since SpartaDos X does not dump you into Basic when it boots up but rather ends up at the DOS prompt, and since it can have another cartridge installed in it, you have two different commands available. The BASIC command will take you to the internal basic on the XE/XL machines. the CAR command will take you to whatever cartridge you have plugged into the SDX cart. In order to return to DOS without losing anything in the editor of whatever language you are using, it is necessary to have a type of MEM.SAV file for each command. These files can be established as part of your CONFIG.SYS file or from the DOS prompt.

The form for this is SET CAR=Dn:filename for the cartridge and SET BASIC=Dn:filename for basic. (The SET command is used to set various environment variables, of which CAR and BASIC are two.) While this feature may seem to be a regression, it is necessary due to the relocatable nature of the DOS.

CAR and BASIC are automatically set to I:CAR.SAV and I:BASIC.SAV when you boot up. You can change these assignments, but you also have the option of deleting them. If these variables do not exist, then the memory save feature will be disabled and either Basic or the cartridge will be entered cold. When calling DOS from either Basic or the cartridge you will be given a warning if the memory save cannot be completed, so you can save whatever you are working on to another disk first.

I must admit that at first I did not care for this feature. However using a ramdisk for this file makes the delay in saving and reading almost unnoticable. It also allows you to go from DOS to internal Basic to a cartridge without losing anything in memory! I have found it to be well worth the small delay it causes.

You can enter the cartridge or internal BASIC with the CAR and BASIC commands. Both these commands allow you to specify a filename with any needed parameters. The binary file so named will be run when the cartridge or Basic is entered. This is how, for example, you would run a compiled Action! program that requires the cartridge library.

Several other environmental variables are available that you can set with the SET command. One of these is the DOS prompt. You can specify the format of your prompt by using certain "meta" characters. The letters L, M, P, D, T, and R, preceded by a dollar sign, will be replaced by the drive Letter or Number, the current directory Path, Date, Time, or a Return. You can also use an underline, which will be replced by a space. For example, "SET PROMPT=$L:" will give you a prompt of A: for drive one. If you are in the GAMES directory on drive one, you could "SET PROMPT=T_DN:)$P)" and your prompt woud look something like this:

```
11:25:42am D1:)GAMES)
```

The normal Dn: prompt is set up with the command "SET PROMPT=D$N:".

PATH can be set to cause other directories to be searched when a command or file is not found in the current directory. Entering PATH with no parameters will show the current search path. Note: the search path is NOT the same as the current working directory!
To see what your current directory is, enter CWD (or CHDIR, or CD) with no parameters. This takes the place of the ?DIR command from earlier versions of SpartaDos.

The FORMAT command presents a menu type screen and allows you to format a disk in any density you want. You can use the high speed skew for the new XF-551 drive, the US skew for the Indus or a doubled 1050, or a standard skew.

You can format the disk in either Sparta mode or Atari mode. The Atari mode allows you to use enhanced density if you should so desire. This formatting menu shows up whenever a format-disk command is issued -- even from within a program!

The FIND command will search all connected drives and all directories for the specified filename. Wildcards are of course accepted. You can limit the search to one drive by including a drive number in the filename.

The keyboard buffer is not loaded as a default. You must select KEY ON to load the file from the cartridge and activate the buffer. Subsequent KEY commands will toggle the buffer off and on.

The LOAD command has the same effect in SDX as in the disk-based version of SpartaDos. This will load a file but not run it. However, you can also use it to load an external command file to keep it resident in memory. The COM files in the cartridge are loaded each time they are called. You could load one or more of them into memory with the LOAD command and they will stay there until you use LOAD with no parameters, which will remove all non-installed programs from memory.

The SWAP command will allow you to swap your drive configuration. SWAP 1,8 would swap drive one with drive 8. SWAP with no parameters will display the current drive map. MIO users should take note that the MIO menu swap configuration must be considered when using this command.

The X command is used to load and run those programs that require all cartridges be removed. Express, Determ, Textpro, Biscom, and the disk-based ARC are examples of programs that must be called using the X command.

A couple of nice features are the PEEK and POKE commands. These work pretty much like the Basic commands except you don't need the comma to separate the address and the value in the POKE command. Also, you can enter the address and value as either decimal or $hex numbers. These two can come in very handy at times, believe me! For example, my AUTOEXEC.BAT file contains the commands POKE 82 0 and POKE $2DA 2 to set the left margin and cursor repeat rate as soon as the system is booted.

Another command that is nice is CLS for clear screen. This is good for some programs that exit to DOS leaving you with a screen full of the program remnants. There is also an RPM command, which allows you to easily check the speed of your disk drives. (The standard speed for an 810 or 1050 drive is 288 rpm. My Happy drive registers about 900 rpm, and an MIO ramdisk clocks in at around 23000!)

The next feature I will cover is one of the most impressive. SpartaDos X contains a licensed ARC program

that will also handle Alf-Crunched files. Here is the format and commands used:

```
SpartaDOS X  ARC ver 1.1

Usage:     ARC cmd[opt] arcfn [flist]

Where "cmd" is one of:
 a   : add file(s) to archive
 m   : move file(s) to archive, then
       delete from current location
 u   : update file(s) in archive,
       keeping the newest versions
       (according to the time/date
       stamp) and adding any source
       files from flist that aren't in
       the archive
 f   : freshen file(s) in archive,
       keeping the newest versions but
       not adding any files that
       aren't already in the archive
 d   : delete file(s) from archive
 x,e : extract file(s) from archive
 p   : print file(s) to screen
 l   : list file(s) in archive
 v   : verbose list of file(s)

Valid "opt" options are:

 b   : retain backup copy of archive
 s   : suppress compression
 w   : suppress warning messages
 n   : suppress notes and comments
 h   : high speed (screen off)
 g   : encrypt/decrypt archive entry
       (followed by password - must
       be the last option, as in ARC
       AMStwister TEST.ARC e,e)

"arcfn" is the archive filename
(drive, path, and ext are optional).

"flist" is the list of files to
extract, add, list, etc. in the
archive. Wildcards are allowed in each
filename. "*.*" is assumed if no
filename is given.
```

This is a fully featured ARC as you can see, and works the same way as the ARC utilities for IBM and ARC.TTP for the ST. It is also extremely fast! The first time I used it to unarchive a file I thought it must not have worked properly since it finished so quickly.

The program will automatically determine whether a file is in ARC or Alf-Crunched format when extracting files, and when compressing will use the most efficient ARC format it can. Anyone who is a frequent downloader of programs that have been compressed will really like having this program always just a command away.

(24)

Most other commands remain the same as they were in the disk-based version of SpartaDos. Some have additional features such as DIR and TYPE. They can each accept some optional parameters. Remember the discussion of the ATR command? The Attributes can be included with these commands. DIR +S will show only subdirectories. DIR +H will show hidden files. You can also use a /P or /C. with DIR. DIR /P will pause the display and wait for a keypress after giving you a full screen of filenames. DIR /C will give you a count of the number of entries in that directory. DIR /PC will do both. TYPE can also use the Attributes and the /P. Also, TYPE is no longer limited to text files with 64-character lines. You can now TYPE any file to the screen.

PRINT has been dropped and the entire I/O re-direction scheme has been changed. Batch files are only read by the command processor and cannot input information into a Basic program since they are no longer system wide. You can divert output of a single command by using >> such as DIR >>PRN:. This would redirect the directory display of drive one from the screen to the printer. DIR >>A:FILENAME would direct the display to the file on drive one. Similarly, you can use <<A:FILENAME to specify input from a file instead of the keyboard.

As I already mentioned, SpartaDos X looks for a file called AUTOEXEC.BAT at boot rather than STARTUP.BAT. This batch file would include any system set up commands you need (like the POKEs mentioned above).

You can execute a batch file from the command processor with the -filename entry. You can also pass parameters to the batch file. As an example let's suppose that you have set up your RIO ramdisk as drive one but that you frequently want to change the file that you are using as your CONFIG.SYS. You have a floppy in drive two that has a library of CONFIG.SYS files named FILE1, FILE2, etc. Here is our batch file, named CHANGE.BAT:

```
COPY %1 %2
COLD
```

Now we would execute this batch file as follows:

```
-CHANGE B:FILE1 A:CONFIG.SYS
```

This would copy FILE1 from your library of CONFIG.SYS files in drive two to drive one and re-boot the system using the new CONFIG.SYS file.

This is not the best example in the world but it does give you an idea of what can be done. You may pass a total of nine parameters to the batch file.

While speaking of batch files and CONFIG.SYS files, a nice feature in creating and editing these files is that

COPY CON: does not clear the screen as COPY E: does. You can TYPE the file out, issue a COPY CON: CONFIG.SYS, move the cursor to the display of the file and edit what you want while hitting RETURN after each line, and then press CONTROL-3 to close the file. This can be a real time saver when setting up various CONFIG.SYS files.

And now a note to TURBO BASIC users. You can now use TB with SpartaDos! However you will need an expanded memory system since you must load the handlers into BANKED memory. That will leave the RAM under the OS available to TB. While I haven't tried it, it might be possible to load the handlers into normal RAM and run TB on a stock XL but it would drastically reduce the memory available for any programs to use.

Well, this has been a brief overview of the SpartaDos X cartridge. In my opinion it is well worth the long wait. While there still could be bugs lurking around in SDX somewhere, if they are there they are extremely subtle. The only problem I can see will be the occasional program that is not compatible with SDX. In the course of beta testing we've found a few such programs, but ICD has quickly remedied this. If you discover a program that doesn't work, report it to ICD and use the disk-based version with it until the next upgrade.

Whatever you do, don't miss this one!

ICD, Inc.
1220 Rock St.
Rockford, IL 61101-1437
info & orders: (815) 968-2228
ICD bbs system: (815) 968-2229

****

STARFLEET MEETING MINUTES

October 14, 1988

Officers present McDaniel, Bender, Guenther, Roubique, Oughton

This meeting at Coco's resturant was back to the environment of the first meeting we had there. The noise level again very acceptiable.
C.J. opened the meeting at 7:20 with some general comments. Remember next month we nominate officers!!! Guy pleaded for volunteers to work Bingo for the club. He discussed the financical advantages to the club of this kind of operation. Charles then gave a state of the treasury report. While debating the site for our next meeting, C.J. demonstrated a slide show of Star Trek images.
After a heated debate on meeting places, We practiced for the first tuesday in next month and voted on a location. The fist vote resulted in seven votes for Coco's, seven for Pomona High school and one "yes". This tie resulted in a re-vote, the second vote was 8 votes for Coco's, 7 for Pomona. Several votes also included other comments: YES, NO, Maybe, and names of presidental candidates.
Chris demonstrated an early design of Atariwriter with Proofreader. As I write this I wish Atari would have released it instead of Atari writer Plus. It would have been like the cartridge version only with a built in spelling checker. The switch from word processor to spelling checker was almost instant and with only a couple of keystrokes. Next, a Demo of Classy Chassy. This is a Pinball game from Clearstar Softechnologies. The most noticable part of this program is the "cheer section", a rather nice looking woman. The advertisment says, "... the better you do, the better the view!" This game allows the player to add english to the ball. For a price of $9.95, if you like pinball, it is a good deal.
The raffle this month was "LORDS OF CONQUEST" from Electronic Arts. This game, for one to four players, is very much like the board game "RISK". There is about 40 different maps to fight over, and included is a editor to create your own map (anyone care to fight over Barsoom?) This game is more than the fight, fight, fight of "RISK", it's main object is not to dominate the world, but to build cities. To build cities raw material is needed, and the raw material comes from the countries you control, or what can be plundered. We give our thanks to Scott Anderson for the demonstration of this program.
To raffle off "LORDS OF CONQUEST" I used the version of "DOORPRIZE" that used the CHEEPTALK speech synthesizer in the January 1987 ANTIC. The curcuit is almost the same as one presented in ANALOG sometime before. The speech is understandable, but not as good as produced by "S.A.M." or COVOX digitized speech.
Various topics were discused before the meeting broke up at about 9:15. Several people stayed to socialize.

NEXT MONTH: AT COCO'S.
To be raffled: PRINT POWER
Agenda item: NOMINATIONS

, except I doubt you'll ever h me at home.

s for programming, I have lately been a bit at a loss for what constuctive programming I could do. If any of you would like to see a program do something... would like to help write a program... etc... please respond in Email on MAAUG-BBS, or write to the above address. I have a vast background in 8-bit Basic and Assembler, so if you have a problem with that... feel free to ask me. I would love to see a MAAUG-sponsored programming project for the 8-bits. (Hopefully, this one won't die like the "Rambo Moose" project did.)

Finally, some sad news for the BBS communities. As I am sure all of you know, MOOSIE has gone on hiatus until such time as he gets the $$ for a new hard drive, and gets the new software written. Madison's OTHER Moose, Robbing Moose, also took down his Bulletin Board System due to problems. This raises the "GREAT" BBS death toll from 1 to 3, with Warlord's Chatline/BBS having gone down about 3 months ago. If you are looking for new places to expand your BBSing horizons, I suggest the Q&A Frame for 8-bit stuff (608) 831-0318, MACC/MIC for ST stuff (608) 263-6057, Atlantis for general discussion (608) 273-9633, and The Party Board for adult discussion (608) 258-9555.

Comments on systems or anything else related to what I've mentioned in this article, or on future content would be greatly appreciated. You can reach me at the above-listed P.O. Box, on the BBS, or at the phone # on the back of your MAAUG newsletter.

<->Tanis<->
SysOp
(608) 274-4256

News from Compuserve

This is an announcement that I found on Compuserve. It looks like someone has plugged a 65816 in their Atari and is getting ready to spread it around. Could be fun! Stay tuned!

PRESS RELEASE

There have been many mentions in the almost decade since the original Atari 400/800 Personal Computers hit the dealers shelves about there being a future upgrade to meet the user's needs, and new and more challenging applications. Finaly that upgrade is available.....The Turbo-816 by DataQue, for the Atari 400/800/XL/XE.

DataQue Software is pleased to announce a powerful new upgrade which was co-designed by Ron Shue, and Chuck Steinman.

This upgrade will be available in two forms. There will be a replacement CPU board for the original 400/800 Computer system, and a plug in module for the XL/XE series. In either case, there usually is no need for any modifications to the existing hardware. The only exception to this is with XL/XE systems which have their CPU soldered inplace, which will require the removal of the existing CPU, and the addition of a standard 40 pin I.C. socket is suggested.

Also included is the Turbo-OS, by DataQue for use with the Turbo-816 CPU boards.The Turbo-816 will not only increase the potential speed of the computer, but also break the 64K memory barrier of the existing systems. Not with the awkward pages memory, but with a fully linear decoded address space of up to 16 megabytes. Benchmarks have put the Turbo-816 into a performance range ABOVE many of the 'other' PCs/!! Special memory boards will be available to take advantage of the new extended addressing range. These will be mounted internal to the coputer cabinet, and in most cases require no hardware modifications. And here is the amazing feature..... While adding all this power and all this expanded addressing, the Turbo-816 for the Atari 8-bit computer systems will maintain compatibility with most currently available commercial and user written software. Using the Turbo-816 even those older programs will enjoy a speed increase!

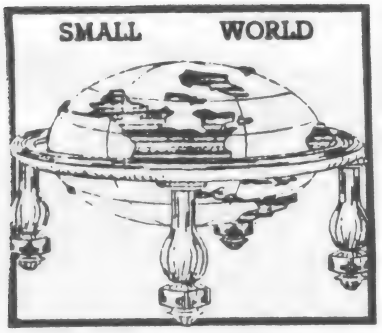
The Turbo-OS is a replacement operating system fo use with the Turbo-816 which will release the 16-bit processor to its full power. Increased speed will be the most obvious change, but hidden in its code, will be an advanced new floating point library that will speed even the original Atari BASIC to new levels of performance. Again, on most systems it will be just a matter of replacing the existing ROM(s) with the Turbo-OS.

The future holds many more products for the Turbo-816 systems including:

1) a real-time multi-tasking operating system kernal
2) a new assembler-editor-debugger package which supports the new assembly level instructions and addressing modes
3) a new BASIC which will speed past the fastest of the current BASICs for the 8-bit machines
4) a new K&R compatible C development package
5) a new Turbo-GOS operating system (graphical based)
6) a developers development kit for new applications

The NEW Atari Turbo-816 should be available by November of 1988. For more information contact your local Atari Dealer or, write:

DataQue Software
Dept. T-816
P.O. Box 134
Ontario, OH 44862



War Inside The Computer

(27)

By: David & Sandy Small

Current Notes 11-88

Introduction

In the past few columns, I've been fairly restrained, a bit formal if you will. Nice little stories about the dark side of bulletin boards. Awards for the development of the Spectre.

While all this mellowness is nice and respectable, it can be a trifle boring.

So, for this column, I'm locking and loading, taking the safety off, switching the selector to "full–automatic", and letting go. Hang on...

What's programming all about?

Some people have this idea of programming as a set of cold, theoretical principles, formed into approved algorithms, daintily laid down in "structured Pascal," and compiled by some poor enslaved microchip somewhere.

That's how programming's taught, anyway. I'm the victim––with a Bachelor of Science (appropriately titled, "B.S.") degree––of a computer science department at a major university.

'taint so. Jack Tramiel says "Business is War". Awww, phooey. Business is giving people something for their money.

The truth is, programming is war!

Let's take for instance my latest little hack, the Spectre 128. I did not idly stroll across a grassy campus on a sunny day, working out algorithms in my mind, sit down in a New Age music–filled room, and touch–type in the code using Pascal. Oh, no.

The development of both the Magic Sac and Spectre more resembled a knife fight. It was me against cold, evil, doubtlessly Communist chips, doing everything in their power not to work.

Dave Slags Pascal

To understand the evil of the Apple ROM chips, you have to first understand Pascal, for the ROMs were written largely in Pascal.

Pascal crawled out from under a rock in the early '70s, and spread like a disease through college campuses, corrupting the minds of thousands of innocents.

The creators of Pascal were appalled by "unstructured languages." To them, the programming world was a jungle. No one followed any "rules." There were no "standards." No one "conformed."

Unstructured languages, like Basic, allowed the programmer to basically do whatever the heck he wanted to do. Go to anywhere in the program, any time. Jump around in clever ways. Get stuff done.

"Oh, fooraw!", said the Pascaldroids. With a true bureaucrat's love of Absolute Control that would make Big Brother cringe, they began a campaign to take away a programmer's freedom of movement.

First came the slogans. "Spaghetti coding!" "Bad programming practices!" "What's important isn't whether or not you get the job done; it's how the code looks!"

And in the greatest snowjob since Lyndon Johnson's campaign ads, the Pascaldroids put across The Big Lie: You should break down every program into a very few control structures. If–then–else. Do–while. And the dreaded GOTO was banned.

More slogans. "Control Structures." "Elegant." And the worst atrocity of all: "Self–documenting code." This one is the worst; it assumes the next person to read your code is a near telepath, who can somehow divine your program's purpose by looking at your code, not your comments.

☞**Small's First Law**: There is no such thing as self documenting code. If you don't have more comments than code, you're blowing it.

What the Pascalzealots never, ever, ever talked about was the layer of separation put between you and the computer when you went to Pascal. The speed of programs dropped, because the Pascal compilers generally created junk code. You never could find out anymore just why the computer crashed anymore, unless by some miracle you found a bug in your Pascal code.

You couldn't just type RUN to find out if your program worked. Oh, no. You had to learn some silly editor, type in your code, then run it through a compiler, then through a code generator, and wait for some slug–like "P–Code Interpreter" to deign to execute your ideas.

Shortly thereafter, Pascal became a religion. Programmers were removed from direct control of their CPU's, their programs were

forced to slow down, and rarely did they know what was going on at the bit level.

Hence, Pascal is the 55 mile per hour speed limit of computer languages.

The same stupid politics apply as with 55. "It's for your own good." "It's for your own safety!"

I heard this Big Lie often through college; I escaped just before Pascal became mandatory. By this time, Pascal had become The Clique in the computer sci department. It was forced on programmers with techniques that would make TV evangelists gag. If you didn't code in Pascal, you were somehow less than macho. If you didn't code in Pascal, you weren't part of the New Wave, you belonged to the past. Why, I had one programmer-to-be tell me that "Programming in Pascal is like wearing a self-imposed straightjacket."

To put this in perspective, imagine if your daughter is going out with a young man tonight. He comes to the door. You welcome him in. You ask what he does; he says, "I wear a self-imposed straightjacket." What tripe.

☞**Small's Second Law**: The truth is, it is possible to write good or bad code in any language. It is possible to apply structure rules in any language.

☞**Small's Third Law**: Any programmer silly enough to straightjacket himself belongs in one.

Only mass conformism gave us Pascal. But soon, the Truth about Pascal began to seep out, in quiet underground movements. Oh, if you dared stand up and say, "Well, I LIKE Basic," you'd be pulled down by the baying Pascal hounds. But evil things were whispered. "Pascal is so inefficient my programs take forever to run." "There are things I can't even DO in Pascal, like mixing types, when I bloody well need to!" "Jeeze, did you see that? My little Pascal program is a 100K long when it's compiled!"

The sad, sad truth was, Pascal was a language with Very Little Mind, to quote Winnie the Pooh. All the religious sayings, all the bleatings about how good it was for you, paled against cold reality: An assembly language programmer could write code that ran circles around Pascal. An assembly language programmer could write code that was vastly smaller than Pascal code. And an assembly language programmer could do many things that Pascal could not even do.

So, the quiet underground movement began, among people smart enough to realize that PascalHype was Hype. For the people unafraid to rock and roll at machine speed, assembler became the way.

- *Lotus 1-2-3* was written in assembler.
- *Wordstar* was written in assembler.
- *dBASE* was written in assembler.
- *XyWrite* was written in assembler.
- Darn near every arcade game was written in assembler.

Am I getting through? *The programs that made people rich were written in assembler.* The programs that screamed with speed were written in assembler. The programs that made you feel your computer was a hotrod, able to leap tall buildings in a single bound, were written in assembler. Those programs SOLD because the users liked the "look and feel" of them. They felt hot and sexy.

Brooke Shields does NOT sell posters because she is Structured. Elegant. And Wearing A Self Imposed StraightJacket.

Don't Look Now, But Here Comes The '80's

Now let's get to the 80's. People started dropping Pascal in droves. The limitations were just too much of a pain. "C", a Pascal language that was actually efficient and fast, and let you do whatever you want, took over, and today is skyrocketing in popularity.

Enter Apple. Apple liked Pascal. Apple liked Pascal A Whole Lot. They made a computer called the Lisa.

Whenever anyone turned on Apple's Lisa machine, whose operating system was written in Pascal, they all had the same comment: What's taking it so long? Why is this thing so slow? And the ultimate: How could they have slowed down a 68000 ***this*** much?

So it came time to do the Macintosh. The drawing routines were written in Pascal, then optimized into assembler where there were bottlenecks. They ended up being 60% of the Mac's ROM (prestored memory). The operating system ended up being written mostly in assembly, for speed.

But the machine, unfortunately, retained a nasty Pascal flavor to it. You had to beg the machine for memory to load your program into. You had to plead with it not to roll your program out of that memory, once you'd gotten it there. Your program would be shifted around in memory underneath you at more or less random times. And all of this is justified in prim, self-righteous tones throughout the Mac documentation as "being good for you."

You couldn't do something easy, like PRINT "HI THERE"; oh, no. First you had to initialize the memory mangler, the heap, QuickDraw, open a GrafPort, set the font, set the font size, set the font characteristics (bold, italics, etc), start up an event handler, read in a

resource with the text "hi there" in it, then, possibly, maybe, you could print "Hi There". It takes nine pages of Pascal to print "Hi There".

You may have heard the Mac is the hardest-to-program machine around; in my humble opinion, that's the Pascal heritage showing through.

Pascal forced programmers to do things just one way, in the idiot control-freak dream of Standardizing Programs, of basically one person saying, "I vant everyone to do it MY WAY!" If you'll recall Nikita Kruschev pounding on the U.N. table with his shoe, you've got a good idea of how standardization committees like to control programmers.

Most of the time, that one way ended up being foolish, ended up making the programs long and slow. The Mac does the same silly thing to programmers.

And this is why every Mac program you read about is "delayed another three months." Or "still pretty buggy." Or "Well, we've dropped the product."

Dave Slags the ROM Chips

So, now, take those ROM chips, with their cold fascist heritage, and plug them into the ST, as I did back in 1985. (It was easy back in '85; they plugged right into the TOS ROM sockets, since there were no TOS ROMS.)

Do you think they appreciated it? Oh, no. They acted exactly like a KGB agent who's been kidnapped and forced to live in the US.

The chips look around the ST bus. Why, there's a good fast hard disk out there. There's a really cool monochrome video display, done properly, not some kludgy way that slows down the processor. There's a fast floppy disk drive.

Do the chips appreciate it? Imagine a KGB agent eying a Kentucky Fried Chicken stand distastefully, and you'll get the idea. The agent is horrified at the lack of standards, the lack of control. Why .. why.. there's not even a dress code here!

The chips declare a jihad; their purpose in life becomes Crashing. In short, they are so unable to adjust to life in the ST that they adopt a mutual suicide pact.

They sit in the computer, plotting far into the night just how they're going to fail. Let's listen in.

"Comrade Lobachevsky! What haff you got?"

"Vell, sir, the disk drives seem to be uncrackable. However, the disk format routine isn't solid, so we could crash there. And there's real possibilities with the serial chips."

So, the next day, someone tries to format a disk in Mac mode. Whammo. The crash page scrolls up, the chips breathe a sigh of relief, and quit.

And later on, someone tries to run a terminal program that goes straight to the serial chip. Crash.

I threatened the chips. I hooked them up to the 110-volt wall outlet through a switch, and said, "Obey, or else!" They sneered, said, "Go ahead and throw the switch, Imperialist! We show you how Pascal Chips die!"

Well, that wasn't going to work. So I started building fences around the chips in software, limiting their freedom to do dirty work.

"Comrade Lobachevsky! Report! You said you were going to crash today!"

"I.. cannot understand it, Comrades. Today the serial port was taken away from us. I tried all the functions, and they all worked."

An evil-looking fellow, Comrade Borodin, speaks up. "I haff a solution. Many progwams accidentally write into location zero. On this machine, that vill wesult in a cwash. Thus, the system will newuhr be stable."

At this, the meeting dissolves into laughter.

Next day, I'm running *Microsoft Excel*. Crash--store into zero.

It looks like I'm stuck. The Motorola manual is quite clear. Storing into zero gives a Bus Error, and you cannot recover from a Bus Error--in fact, you need to get a 68010 chip to be able to recover. (All you hot computer sci types--that's so you can implement virtual memory, paging in off disk.)

So, I neatly close the Motorola book, and go try to recover from a Bus Error. Why, golly, the book was wrong. I go ahead and write some software, and ignore the store into zero. It isn't easy, but it works.

The ROMs are in deathly horror now. "Comrade Borodin! Ve haff stored into location zero fifty times today, and there are no crashes! Guard! Take Borodin to the firing squad!"

(Sound of AK-47 fire).

And so it goes. As the chips discover another way to die, I fence that way off. As Mac programs do one knuckle-headed trick after another, create problems that crash even real Macs, I trot around behind them with my pooper-scooper, picking up the debris.

With the Spectre 128, there are demilitarized zones, fences, landmines, tripwires, booby traps, warnings, and more. The ROMs are in a prison camp pretty much. Sure, evil plans are still being hatched; they've found some way to mess up "Page Preview" on *Microsoft Word 3.02*. (128K ROMs only). And sometime soon, I'll find out how they did it, and drive a stake through it's heart.

But it's been a struggle the whole way, a war. I've lost a lot of battles, and there's still a few things that won't ever work--Mac MIDI, for instance--but overall, I've won the war.

Conclusion

So there you have it. A brisk discussion of Pascal that will make the blood pressure spray out the

ears of any Pascal lover. The real grim truth of what those Apple ROMs talk about at night. And a look into the bizarre, twisted psychology of an assembly language programmer.

If you'd care to comment on this column, and drop me a note online, follow the following program to end up with my address.

```
IF {LANGUAGE(Favorite) == PASCAL} &AND&
   {EMOTION = OUTRAGE}
     DO
          Compuserve address = 00000,000;
          GENIE address = Nil;
          Usenet Address = ping!pong;
ELSE
          Compuserve address = 76606,666;
          GENIE address = DAVESMALL;
          Usenet Address = (whatever)!well!dsmall;
ENDIF
```

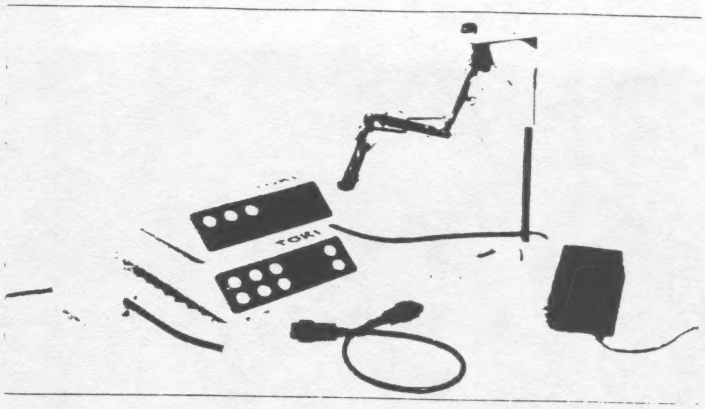
DH-101 Robot

The future of robotics is very. very quiet...

The DH-101 Robot Arm elegantly demonstrates the astounding capabilities of shape-memory alloys. The five-action arm moves entirely by the contraction of tiny Biometal wires, which actually shorten in length when electrically powered.

The DH-101 can be operated in many ways: under direct push-button control by joystick, through the replaying of "learned" sequences, and under your computer's control through a parallel printer port or the unique "E-Z Link" Optical Serial Interface which accepts graphic commands from almost any computer monitor.

The Robot's sophisticated design includes molded and machined parts, as well as red LEDs and a blue light in front dome that add a futuristic accent. The DH-101 makes an eye-opening display item, and a great tool for any robotics experimenter or researcher.

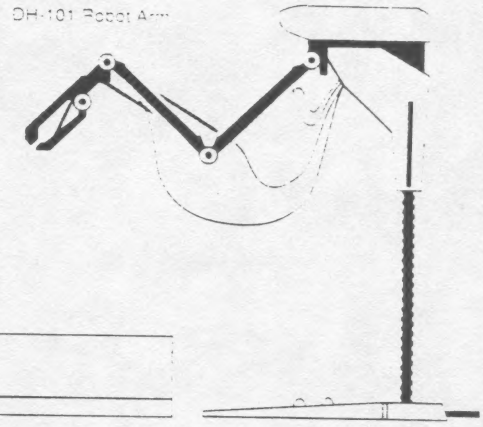


COMPLETE ROBOT PACKAGE - Includes all items above and owner's manual. Operate DH-101 Arm directly, in auto modes, via joystick (not included), and from almost any personal computer.
3-004 DH-101 Complete Robot Package $425.00

STARTER PACKAGE - Includes DH-101 Robot Arm, Driver Box, AC Adapter and manual. Operate DH-101 directly or via your PC.
3-010 DH-101 Starter Package $299.00

Hard Disk Info Unrelated

The AVERAGE ACCESS TIME in ms (milli seconds) as specified by the drive manufacturer is always meant for the whole drive capacity. That means that two drives with the same average access time but different capacities, let's say 20 and 40 MB are in reality totally different in actual access time, the higher capacity drive will be in most cases roughly double as quick! Since the ATARI allows you only a maximum of about 18 MB per partition, actual average access times are much better with high capacity drives! Lets look as an example at a SEAGATE 251. This drive has an av. access time of only 8 ms over the whole capacity of 42 MB, a track to track access time of 25 ms average. In general with a bigger partition you can expect something like 25 ms average access time will be accordingly better. Depending on the encoding scheme (MFM or RLL) a track has either 17/18 or 25-28 sectors. A 1:1 interleave means that every 2nd sector is skipped to get consecutive physical sectors, an interleave of 2 means 2 sectors are skipped and so on. After reading a sector the controller needs a little of time to digest the information, with an interleave factor of 1 there is a bigger gap between sectors than with a higher interleave factor since the following skipped sectors supply additional time for processing the information just read.
The tradeoffs are then quicker information flow with interleave 1 but a little bit less storage capacity due to the increased gap between sectors. In MFM encoding you have 17 versus 18, in RLL 25 against 26 or 27, resulting in a roughly 5% storage loss.
All our kits run with a 1:1 interleave, the supplied software allows you to choose higher interleaves as well and decide for your own tradeoffs. Most of our customers so format with an interleave of 1.
To put things in perspective: A PC-clone runs typically with an interleave of 6 or 7, an AT-clone with 3, only an AT with 16-bit controller card can run an interleave of 1!



DH-101 Robot Arm - ...ist, elbow, grip and three-action shoulder move with life... ...metal shape memory alloy wires. Clean, futuristic design. ...equires Driver Box and AC Adapter.

Height: 240mm
Arm length: 160mm
Cord length: 600mm

Driver Box - Power and contr... Robot Arm directly, through Memory Box and Interface Box, or via joystic... as Radio Shack #270-1701 (not included). Has internal NiCad battery pack for portability.

Dimensions: 185 x 130 x 45mm
Power: 9 Volts, 1.2 Amp
NiCad battery pack: 7.2V, 1.2 AH

Memory and Interface Box - Expand your possibilities. Run the DH-101 Robot in auto modes, teach and playback over 1000 moves, interface it to your computer through Centronics-type parallel port, and E-Z Link serial interface.

Dimensions: 185 x 130 x 45mm
Power supplied by Driver Box.

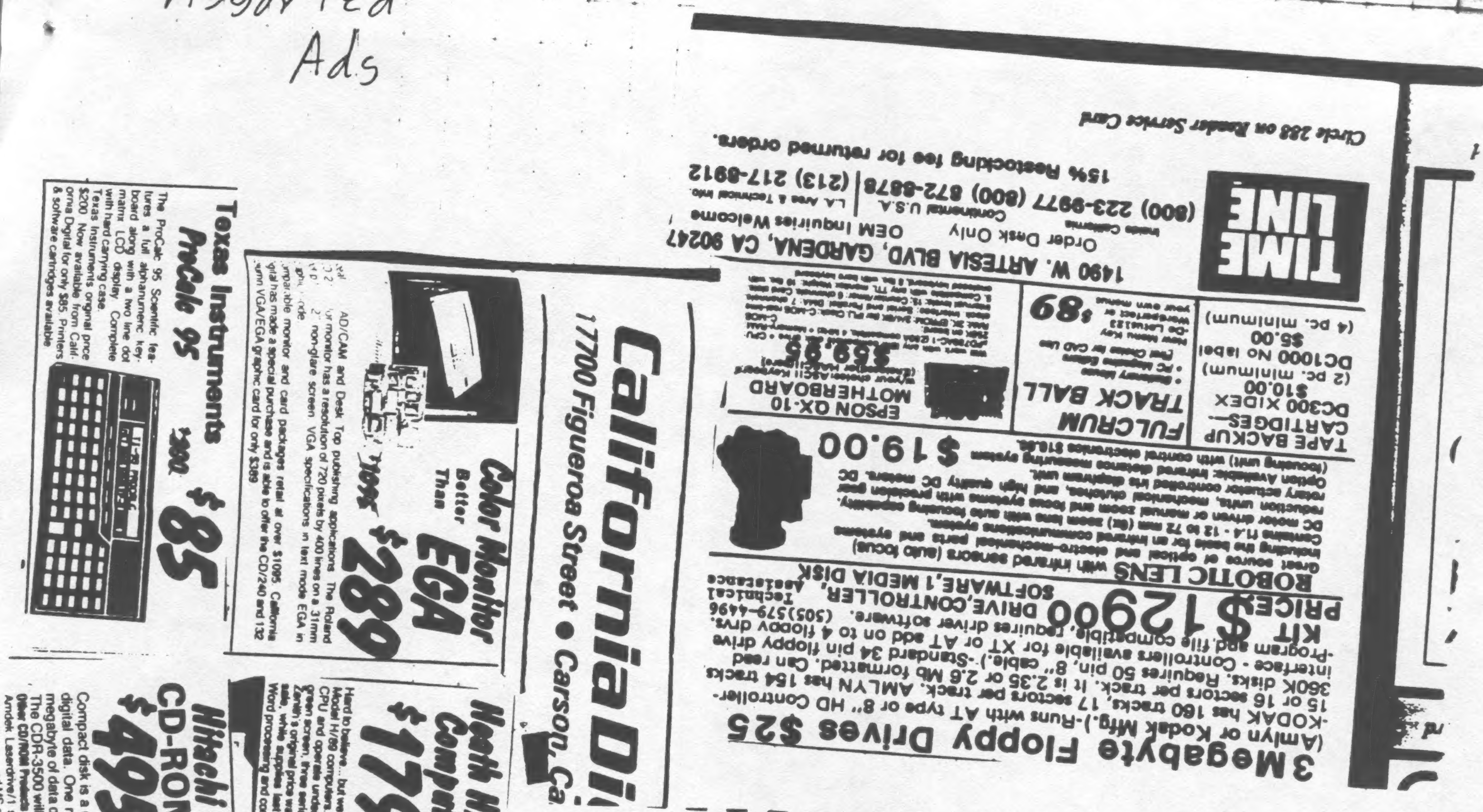
AC Adapter - Plugs into wall outlet to run DH-101 Robot Arm, Driver and Memory Boxes. Only one required per system. 117 VAC, 60 Hz input.

Dimensions: 60 x 50 x 42mm
Input: 117 VAC, Output: 9 VDC

E-Z Link Optical Serial Interface - Control your DH-101 Robot from almost any computer. Unique universal interface mounts on monitor and receives graphic signals. Requires Memory and Interface Box and one 9 Volt battery.

Dimensions: 60 x 50 x 22mm
Flex stem length: 145mm
Cord length: 800mm

Assorted Ads



DAT adapted for data

At first glance it seems unlikely that a technology conceived for car and home stereos and boom boxes would find application in the weightier world of minicomputers and workstations.

But digital audio tape (DAT), announced by the Japanese a couple of years ago, is now being adapted for use in computer storage. Due out sometime next year, computer tape drives that accept cassettes about the size of a U.S. business card, holding a tape 4 millimeters wide, are expected to find immediate use in computer workstations and minicomputers and, as the price comes down over the next few years, in personal computers and portables, too.

DAT has met resistance in the United States from recording artists and record companies who fear it might hurt the booming compact disc business—DAT drives can record as well as play back.

Japanese manufacturers have voluntarily withheld DAT products from the U.S. and some European countries to avoid conflict. However, the drives and prerecorded DAT tapes have been on the market in Japan, West Germany, and France for about a year.

Hewlett-Packard Ltd., Bristol, England, and Sony Corp., Tokyo, have drawn up technical standards for the DAT mass-storage drives, to be called digital data storage (DDS). So far, seven other tape drive manufacturers, including NV Philips Gloeilampenfabrieken, Eindhoven, the Netherlands, and Aiwa Co. Ltd., Tokyo, have agreed to go along with the DDS standards.

The first DDS drives will b

Computers take DAT

about the size of a 5-1/4-inch floppy-disk drive. Each cassette will store 1.3 gigabytes of data, a huge increase on the 150-megabyte capacity of the conventional 1/4-in.-wide cassette tapes now used for computer data storage, although the next generation of these is expected to hold 320 megabytes.

Conventional digital mass-storage tape drives store data in tracks lengthwise along the tape with guard bands in between. But to achieve higher density, DAT drives deposit information in overlapping diagonal lines in what is known as a helical-scan method.

"With DAT, you can get a very high storage density on a very inexpensive medium," Bert Vermeulen, program manager for DAT development at HP in Bristol, told THE INSTITUTE.

Current DAT formats

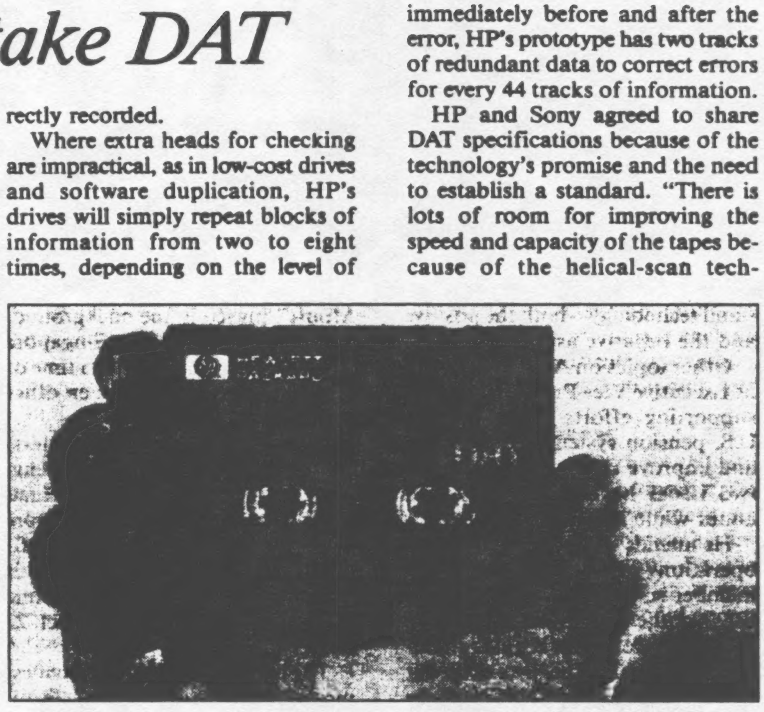
The new DDS products will incorporate the formats and standards of current DAT consumer products, but with modifications to achieve the greater reliability required. For instance, HP's prototype achieves high density with the same helical-scan recording technique, but has an extra pair of magnetic heads that follow the two recording heads to check that the data are correctly recorded.

Where extra heads for checking are impractical, as in low-cost drives and software duplication, HP's drives will simply repeat blocks of information from two to eight times, depending on the level of reliability needed. That procedure, of course, slows information retrieval and reduces the density of data on the tape.

In addition, the computer-storage products are being built with more rigorous error-correcting codes. Whereas standard DAT corrects errors by interpolating from the values immediately before and after the error, HP's prototype has two tracks of redundant data to correct errors for every 44 tracks of information.

HP and Sony agreed to share DAT specifications because of the technology's promise and the need to establish a standard. "There is lots of room for improving the speed and capacity of the tapes because of the helical-scan technique," said Vermeulen.

DDS computer drives, he said, may even eventually replace nine-track tapes in some installations. "Of course, people have been predicting the obsolescence of nine-track tapes for 20 years," he said, "so it may take more than a decade." —*Fred Guterl*



Carl Russo

Hewlett-Packard and other companies plan to introduce tape drives this year that will store computer data on the same kind of 4-millimeter tape already developed for digital audio tape (DAT) recording.